UNIVERSITY OF HELSINKI

FACULTY OF SCIENCE

THEORETICAL AND COMPUTATIONAL METHODS

Image Deblurring: Comparing the Performance of Analytical and Learning Methods

MASTER'S THESIS

Author:
Sangita Seshadri

Supervisor:
Prof. Samuli Siltanen

June 1, 2020

HELSINGIN YLIOPISTO — HELSINGFORS UNIVERSITET — UNIVERSITY OF HELSINKI

Tiedekunta/Osasto — Fakultet/Sektion — Faculty	Laitos — Institution — Department
Faculty of Science	Theoretical and Computational Methods

Tekijä — Författare — Author
Sangita Seshadri

Työn nimi — Arbetets titel — Title

Image Deblurring: Comparing performance of Analytical and Learning Methods

Työn laji — Arbetets art — Level	Aika — Datum — Month and year	Sivumäärä — Sidoantal — Number of pages
Master's Thesis	May 2020	51

Tiivistelmä — Referat — Abstract

Blurring is a common phenomenon during image formation due to various factors like motion between the camera and the object, or atmospheric turbulence, or when the camera fails to have the object in focus, which leads to degradation in the image formation process. This leads to the pixels interacting with the neighboring ones, and the captured image is blurry as a result. This interaction with the neighboring pixels, is the 'spread' which is represented by the Point Spread Function. Image deblurring has many applications, for example in Astronomy, medical imaging, where extracting the exact image required might not be possible due to various limiting factors, and what we get is a deformed image. In such cases, it is necessary to use an apt deblurring algorithm keeping all necessary factors like performance and time in mind. This thesis analyzes the performance of learning and analytical methods in Image deblurring Algorithm.

Inverse problems would be discussed at first, and how ill posed inverse problems like image deblurring cannot be tackled by naive deconvolution. This is followed by looking at the need for regularization, and how it is necessary to control the fluctuations resulting from extreme sensitivity to noise.

The Image reconstruction problem has the form of a convex variational problem, and its prior knowledge acting as the inequality constraints which creates a feasible region for the optimal solution. Interior point methods iterates over and over within this feasible region. This thesis uses the iRestNet Method, which uses the Forward Backward iterative approach for the Machine learning algorithm, and Total Variation approach implemented using the FlexBox tool for analytical method, which uses the Primal Dual approach.

The performance is measured using SSIM indices for a range of kernels, the SSIM map is also analyzed for comparing the deblurring efficiency.

Avainsanat — Nyckelord — Keywords
Deblurring, Interior Point Methods, Primal Dual Method, iRestNet

Säilytyspaikka — Förvaringsställe — Where deposited

Muita tietoja — Övriga uppgifter — Additional information

Contents

List of Figures

List of Tables

Introduction

The goal of this Thesis is to analyze the performance of learning and analytical methods in Image deblurring Algorithm. Blurring is a common phenomenon during image formation due to various factors like motion between the camera and the object, or atmospheric turbulence, or when the camera fails to have the object in focus, which leads to degradation in the image formation process. This leads to the pixels interacting with the neighboring ones, and the captured image is blurry as a result. This interaction with the neighboring pixels, is the 'spread' which is represented by the Point Spread Function. Image deblurring has many applications, for example in Astronomy, medical imaging, where extracting the exact image required might not be possible due to various limiting factors, and what we get is a deformed image. In such cases, it is necessary to use an apt deblurring algorithm keeping all necessary factors like performance and time in mind.

We will first go through the Inverse problem, and how ill posed inverse problems like image deblurring cannot be tackled by naive deconvolution. We also look at the need for regularization, to control the fluctuations resulting from extreme sensitivity to noise.

Interior point methods are then introduced, which are used in both the algorithms. The Image reconstruction problem has the form of a convex variational problem, and its prior knowledge acting as the inequality constraints which creates a feasible region for the optimal solution. Interior point methods iterates over and over within this feasible region. We consider the iRestNet Method [14], which uses the Forward Backward iterative approach for the Machine learning algorithm, and Total Variation approach implemented using the FlexBox tool [5] for analytical method, which uses the Primal Dual approach.

The results for both these methods are compared using the Structural Similarity Measure, which considers all the parameters sensitive to human eye, and how it perceives an image. Though both these methods have their own advantages and disadvantages, in terms of time to train a new kernel, or finding the correct regularization parameter, we see that in terms of performance and accuracy of the deblurred resultant image, the deep learning method performs quite well compared to the Total Variation Method.

1. Inverse Problem

A direct problem is the one where we get data from a known object. Inverse problem, as the name suggests is reverse of that. We have the data, and aim to get the original object from the data. X Ray tomography is one interesting example of such a problem, where the direct problem is to determine the X Ray data from a known structure, and it's inverse problem is recovering the structure from the X-Ray data measurement. Image deblurring, which is what we will focus on, is also a similar inverse problem in which we have a blurred image, and we want to recover the original sharp image.

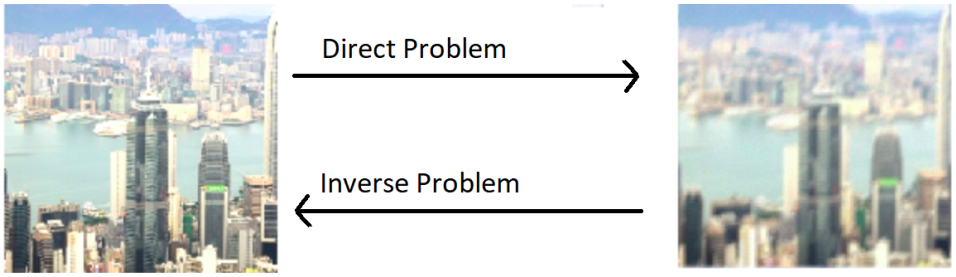

Figure 1.1: The inverse problem known as Deblurring

1.1 Convolution and Deconvolution

The direct problem of representing the blurry image from a sharp one is mathematically represented by Convolution of the sharp image with the point spread function. It represents practical measurements which are affected by noise and other perturbations due to the limitations of the measurement process. The corresponding inverse representation of this process to get the sharp image from blurred one is Deconvolution.

Definition 1.1.1. The convolution of two functions f and p is defined as:

$$(p * f)(x) = \int f(x - x')p(x')dx' \tag{1.1}$$

This transforms the function f into a new one. The change in f depends on p, where the values of p at each point functions as weights and decides the change in f. As defined in equation 1.1, the function f is shifted for each point x and then given the corresponding weight from p.

In discrete model, the convolution between f and p, is defined as:

$$(p * f)_j = \sum_{i=-k}^{k} p_i f_{j-i} \tag{1.2}$$

where $f \in R^n$, $p \in R^{2k+1}$ and $(p * f) \in R^n$.

Figure 1.2 shows the signal f convolved with signal $p = [1, 2, 3, 4, 5]$ after normalization. The convolved signal is smoother, which is what the blurring effect of the Point spread function (p) is.

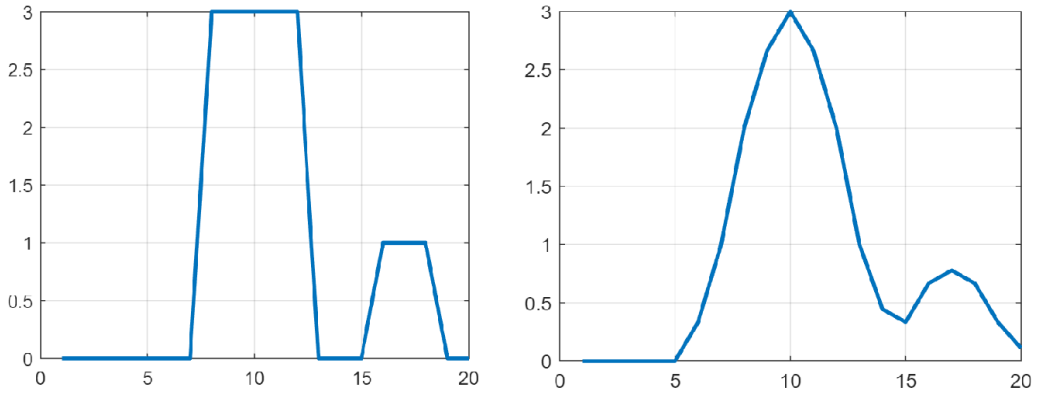

Figure 1.2: Left: The function f, Right: The convolved function $(p * f)$

Now equation 1.2 can be represented in the form of a convolutional matrix Af. Let's expand equation 1.2 to get the matrix A. Considering $k = 2$, and $n = 6$, we have $p = [p_{-2}, p_{-1}, p_0, p_1, p_2]$, and $f = [f_1, f_2, f_3, f_4, f_5, f_6]$

$$\begin{aligned}
(p * f)_1 &= p_{-2}f_3 + p_{-1}f_2 + p_0 f_1 + p_1 f_0 + p_2 f_{-1} \\
(p * f)_2 &= p_{-2}f_4 + p_{-1}f_3 + p_0 f_2 + p_1 f_1 + p_2 f_0 \\
(p * f)_3 &= p_{-2}f_5 + p_{-1}f_4 + p_0 f_3 + p_1 f_2 + p_2 f_1 \\
(p * f)_4 &= p_{-2}f_6 + p_{-1}f_5 + p_0 f_4 + p_1 f_3 + p_2 f_2 \\
(p * f)_5 &= p_{-2}f_7 + p_{-1}f_6 + p_0 f_5 + p_1 f_4 + p_2 f_3 \\
(p * f)_6 &= p_{-2}f_8 + p_{-1}f_7 + p_0 f_6 + p_1 f_5 + p_2 f_4
\end{aligned}$$

The remaining values in f : f_{-1}, f_7 and f_8 are taken to be zero, which is called zero padding. On substituting these are zeros, we get the convolutional matrix as:

$$A = \begin{bmatrix}
p_0 & p_{-1} & p_{-2} & 0 & 0 & 0 \\
p_1 & p_0 & p_{-1} & p_{-2} & 0 & 0 \\
p_2 & p_1 & p_0 & p_{-1} & p_{-2} & 0 \\
0 & p_2 & p_1 & p_0 & p_{-1} & p_{-2} \\
0 & 0 & p_2 & p_1 & p_0 & p_{-1} \\
0 & 0 & 0 & p_2 & p_1 & p_0
\end{bmatrix}$$

The convolutional model then can be represented as:

$$Af = m \tag{1.3}$$

We can extract the original signal f by naive deconvolution as:

$$f = A^{-1}m \tag{1.4}$$

This naive deconvolution approach doesn't work on real data, which usually have noise due to external perturbations, and that causes the inverse problem to be ill posed. An ill posed problem implies that even small fluctuations and noise in the data measurement can lead to large errors, and what we get as a result is significantly different from the original object.

Definition 1.1.2. Well Posedness [15]

Hadamard introduced the concept of well posed problem as the one which satisfies the following condition:

1. Existence: There should be atleast one solution.
2. Uniqueness: There should be atmost one solution.
3. Stability: The solution must depend on the continuity of the data.

If any of the three properties isn't satisfied, then the problem is ill posed.

In the frequency domain, the Fourier transform of the data (blurred image) is given as:

$$F(m) = F(PSF) * F(f)$$

where F denotes the Fourier transform. This is derived by taking the Fourier transform of the convolution operation. This gives us the original function in frequency domain as

$$F(f) = \frac{F(m)}{F(PSF)}$$

In case of discrete signal, we can write this as:

$$F_k = \frac{M_k}{PSF_k} \; k = 0, 1, ..., N-1$$

For some frequencies, the PSF could have extremely small values. In such cases, if there is noise in the signal $m(t)$, it will be amplified to a large extent in the Fourier domain when divided by such small values of $F(PSF)$. The resultant signal we get then is completely unrecognizable since even the slightest noise blows up.

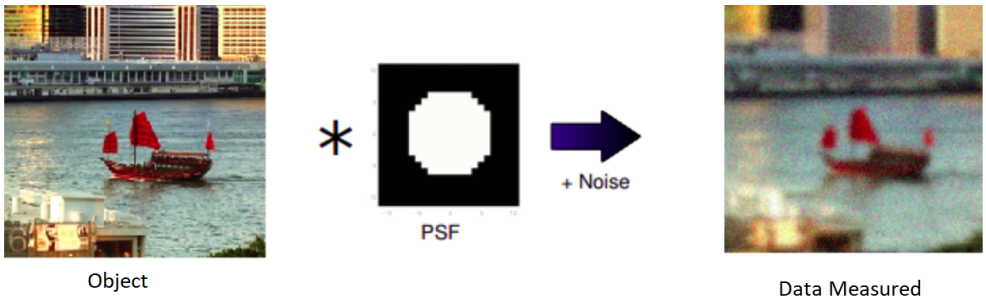

Figure 1.3: How real data is impacted during measurement

1.2 Image Reconstruction

1.2.1 Mathematical Modeling of Image reconstruction problem

While recording an image, the neighbouring pixels can impact each other, which is what the PSF represents, and that causes the blur in the resulting image. The blurriness is caused by the process of image creation, and could be due to the defocus of the lens or the motion of the camera. This is a deterministic process, since we know the direct relationship between the original and blurred image through the PSF, and can be accurately described as a mathematical model.

Noise, on the other hand is the degradation of the created image introduced due to measurement errors or other perturbations in the detection. This is a statistical process. Taking this into account, we can write the image measurement as:

$$m = D'(Af) \tag{1.5}$$

where $m \in R^m$ is the observed data, $f \in R^n$ is the actual data, $A \in R^{m \times n}$ is the PSF or the blurring operator, and D' is the noise perturbation operator.

The PSF determines how each pixel is blurred. We assume that the PSF is space invariant, which means irrespective of the location of the point source, the PSF is the same. The PSF, then need not be the same dimension as that of the image.

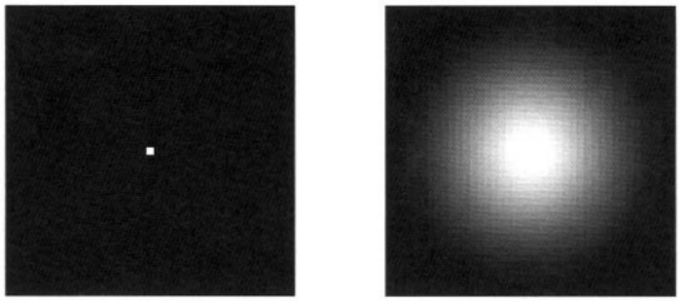

Figure 1.4: Left: A point source. Right: The blurred point source, called point spread function. Image from [10]

Let's consider additive noise. In this case, we have the representation of the blurred image as:

$$m = Af + \epsilon$$

where A is the point spread function matrix and f is the original signal, m is the measured signal and ϵ is the noise. The naive reconstruction attempt can be formulated as:

$$f \approx A^{-1}m \approx A^{-1}(Af + \epsilon) = f + A^{-1}\epsilon$$

In case of zero noise, we get an accurate reconstruction of the signal. However, doing so is an inverse crime and doesn't represent the real measurement. When even a little noise is added to it, as shown in Figure 1.6, the recovered signal from naive reconstruction is completely different from the original (Figure 1.5). This shows how even the smallest errors in measurement, impact the recovered signal to a large extent. In order to overcome this extreme sensitivity, Regularization technique is used to recover the signal.

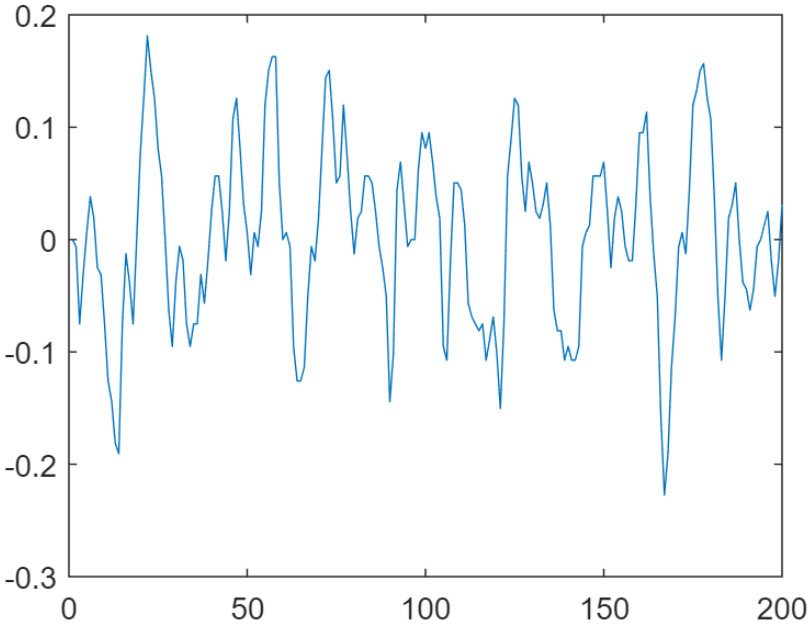

Figure 1.5: The original signal f which we will try to recover in Figure 1.6

1.3 Ill-posedness

From graphs of section 1.2.1, we can notice that even small disturbances gave an unstable result. Ill-posedness of the inverse problem leads to such instability.[15]

Let's consider a forward map $A : D(A) \to Y$ where $D(A) \subset X$ and X and Y are Hilbert spaces. X is the model space and Y is the data space. This forward map is the representation of the mathematical form

$$m = Af + \epsilon$$

where $f \in D(A)$ is the original signal we want to extract, $m \in Y$ is the measured data and ϵ is the noise satisfying $\|\epsilon\|_Y \leq \delta$ and $\delta > 0$

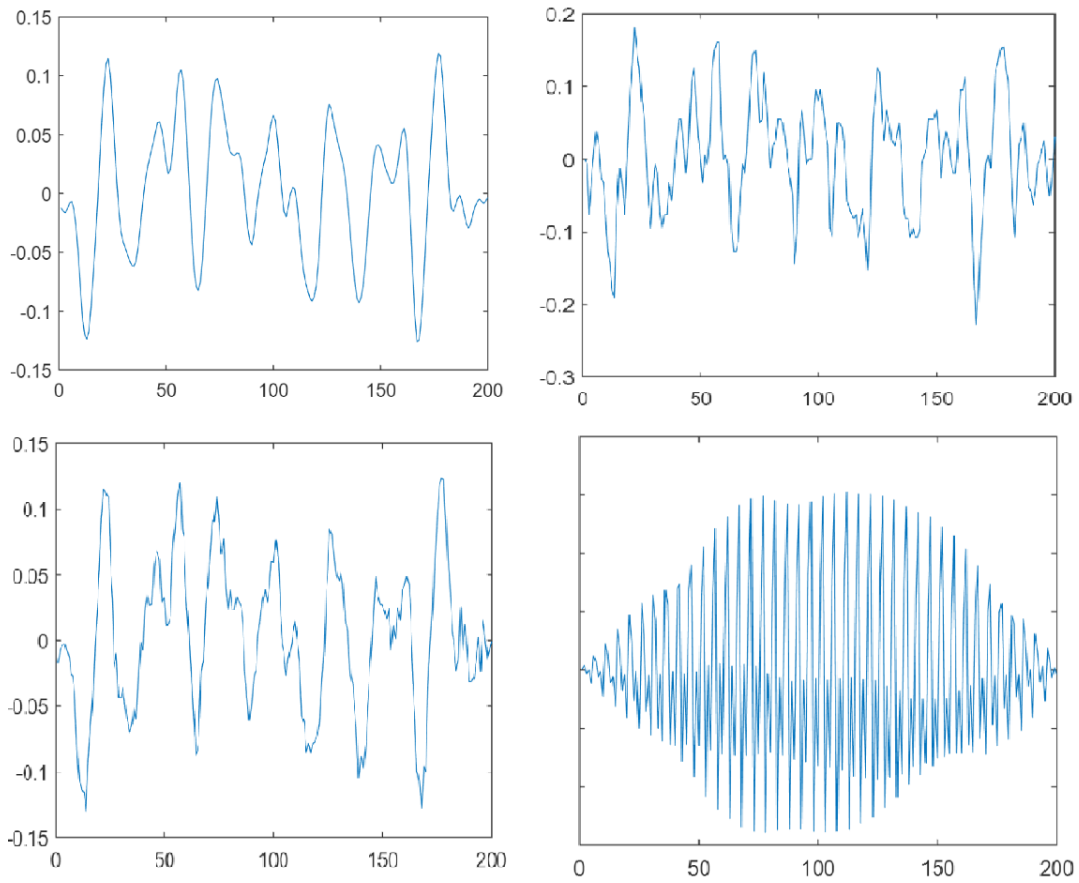

Figure 1.6: Top Left: Data m in case of no added noise, Top Right: Naive reconstruction (with inverse crime) from noise free data, Bottom left: Slightly noisy data m, Bottom Right: Naive reconstruction with noisy data

Now in case of a well posed problem, the forward map is bijective from X to Y and has a continuous inverse A^{-1}

In case of ill posed problem, the condition of Existence is violated if the measured data $Af + \epsilon \notin A(D(A))$. The Uniqueness condition fails when two quantities $f_1, f_2 \in D(A)$ gives the same result $Af_1 = Af_2$. The third condition of stability, does not hold as the forward map does not allow for a continuous inverse.

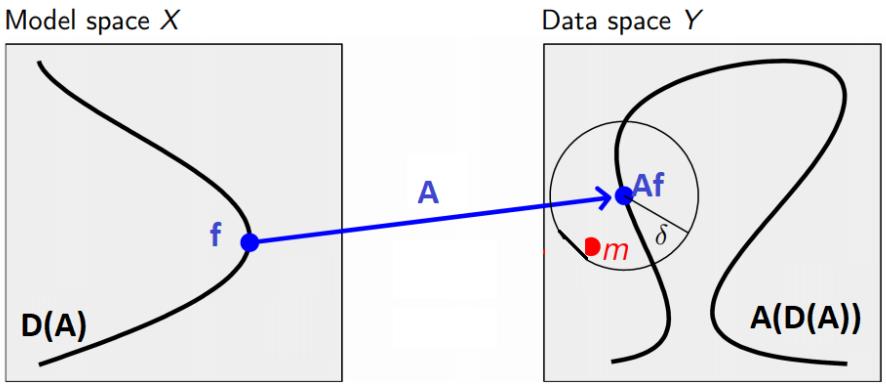

Figure 1.7: Forward map A

1.4 Regularization

Regularization is the technique we use to deal with the ill posedness of the inverse problem, to control its extreme sensitivity to noise.

[9] describes regularization as the approximation of an ill posed problem by a family of well posed problem.

The inverse problem is $f = A^{-1}m$.

In ill-posed inverse problems, there is no continuous function from the data space Y to the model space X, that would map $Af \in Y$ to $f \in X$.This is a result of extreme sensitivity to even small measurement noise in Af.

Now, we don't have information on the exact data 'm', we know the approximation m^δ with:

$$\|m^\delta - m\| \leq \delta$$

where m^δ is the noisy data and δ is the noise level. Now since m and Af are close to each other, it is safe to assume that $A^{-1}Af$ and $A^{-1}m$ are close to each other as well. However, this approach fails for ill posed problems, since A might not be injective or surjective and even if its inverse exists, it might not be continuous. [15]
Let's consider an approximate value of f as f_α^δ which depends continuously on the perturbed measurement m^δ, and as the noise level $\delta \to 0$, then $f_\alpha^\delta \to f$, for the appropriate value of the regularization parameter α.

We then replace a regularized A^{-1} by R_α, to get:

$$f_\alpha^\delta = R_\alpha y^\delta$$

[15] defines regularization strategy and admissible choice of regularization parameter as:

Definition 1.4.1. Let X and Y be Hilbert Spaces. Let $A : X \to Y$ be an injective bounded linear operator. Consider the measurement $m = Af + \epsilon$. A family of linear maps $R_\alpha : Y \to X$ parameterized

by $0 < \alpha < \infty$ is called a regularization strategy if

$$\lim_{\alpha \to 0} R_\alpha A f = f$$

for every $f \in X$.

Further, assume we are given a noise level $\delta > 0$ so that $\|m - Af\|_Y \leq \delta$. A choice of regularization parameter $\alpha = \alpha(\delta)$ as a function of δ is called admissible if:

$\alpha(\delta) \to 0$ as $\delta \to 0$, and

$\sup_m(\|R_\alpha(\delta)m - f\| : \|Af - m\| \leq \delta) \to 0$ as $\delta \to 0$ for every $f \in X$

1.4.1 Total Variation Regularization

[15] defines Total Variation of a function as:

Definition 1.4.2. Let f be a real-valued function defined on interval [a,b]. The total variation of f is:

$$TV(f) = \sup \sum_{i=1}^{k} |f(x_i) - f(x_{i-1})|$$

where the supremum is over all partitions $a = x_0 < x_1 < x_2 < .... < x_k = b$ of $[a, b]$.

If f is differentiable, we can modify the result as a differential of f over the entire range $[a, b]$. Taking $\Delta x_i = x_i - x_{i-1}$, and $\Delta x_i \to 0$, we get,

$$TV(f) = \sup \sum_{i=1}^{k} \frac{|f(x_i) - f(x_{i-1})|}{|x_i - x_{i-1}|} |x_i - x_{i-1}|$$

which can be written as:

$$TV(f) = \int_a^b |f'(x)| dx$$

We can write this equation for higher dimensions as:

$$TV(f) = \int_\Omega |\nabla f(x)| dx$$

where $\Omega = [a, b]^n$.

Total variation regularization for the measurement $m = Af + \epsilon$ is:

$$T_\alpha(m) = \arg\min_{z \in R^n}(\|Az - m\|_2^2 + \alpha \|TV(f)\|)$$

$$T_\alpha(m) = \arg\min_{z \in R^n}(\|Az - m\|_2^2 + \alpha \|Lz\|)$$

where L is the discrete differentiation matrix, and α is the regularization parameter.

Total Variation Regularization computes the balance between the following requirements:

1. $T_\alpha(m)$ should give a small residual $AT_\alpha(m) - m$.
2. $LT_\alpha(m)$ should be small in l^1-norm.

2. Interior Point Methods

Interior point methods are iterative algorithm for constrained optimization problems, where we want the solution to lie in the interior of the feasible region. Appendix A explains about the complex optimization problem in more detail. The iRest Deep learning method deployed uses the Forward backward interior point method, and the analytical method of Total Variation is using the Primal Dual interior point method.

2.1 Statistical Representation of the Image reconstruction problem

Let us first represent the image reconstruction problem, from the statistical properties, before applying the interior point algorithm. For this, we will first compute the maximum likelihood estimate which gives us the most probable result. Now, since we have a prior distribution, which is the known properties of the image, we then use the Bayesian method to compute the conditional probability with the prior, which is the Maximum a posteriori method.

2.1.1 Maximum A Posteriori Approach

We can express each pixel m_i of the measured data in equation 1.5 as a realization of random variable M_i, due to the introduction of noise D'. The probability density of random variable M is denoted as $p_M(m; f)$ due to its dependence on f.

The reference [3] defines the likelihood function as :

$$L_m^M(f) = p_M(m, f)$$

such that

$$f^* = \arg\max_{f \in R^n} L_m^M(f)$$

so this observed data becomes the most probable.

Now, if we also assume the unknown data f is the realization of vector-valued random variable F, and is denoted by $p_F(f)$. This is called a prior, which is some known properties of the data.

The probability density $p_M(m; f)$ can then be represented as a conditional probability density of M.

$$p_M(m|f) = p_M(m|F = f) = p_M(m; f)$$

We can then compute the conditional probability density by applying the Bayes Theorem [11].

$$p_F(f|m) = \frac{p_M(m; f)p_F(f)}{p_M(m)}$$

Which gives us the posterior probability density of F.

$$P_m^F(f) = p_F(f|m) = L_m^M(f)\frac{p_F(f)}{p_M(m)}$$

We can now define the maximum a posteriori (MAP) of the unknown measurement f as any f^* which maximizes the posterior probability $P_m^F(f)$, such that

$$f^* = \arg\max_{f \in R^n} P_m^F(f)$$

Assuming prior of Gibbs type:

$$p_F(f) = \frac{1}{Z}e^{-\lambda\Omega(f)}$$

where Z is the normalization constant, λ is positive parameter, and $\Omega(f)$ is a convex functional. Now, we can apply the negative logarithm, to change it to a minimizational problem as:

$$f^* = \arg\min_{f \in R^n} J(f, m)$$

where

$$\begin{aligned} J(f, m) &= -A\ln(P_m^F(f)) + B && (2.1)\\ &= -A\ln(L_m^M(f)) - A\ln(Z) - A\ln(p_M(m)) + \lambda A\Omega(f) + B && (2.2)\\ &= f'(f, m) + \lambda R(f) && (2.3) \end{aligned}$$

$R(f)$ is the regularizational term, and λ is the regularization parameter.

2.2 Barrier Function

The image is represented as:

$$m = D(Af)$$

where $m \in R^m$ is the blurred and noisy image, $f \in R^n$ is the original image and $A \in R^{m\times n}$ is the PSF matrix/blurring operator and D is the noise perturbation operator.

From MAP approach, we get the constrained minimization problem:

$$\min_{f \in R^n} f'(Af, m) + \lambda R(f) \tag{2.4}$$

We define a Barrier function $B(x)$, which prevents the function from leaving the feasible region during minimization.

$$B(x) = -\sum_{i=1}^{p} \ln\left(c_i(x)\right)$$

where x lies in the feasible domain, p is the number of inequality constraints, and c_i are the inequality constraint functions.

The p inequality constraints define the feasible set (see Appendix A), and in terms of image reconstruction problem, these constraints denote the properties of priori of the image.

Now, the problem is to minimize the resultant function which includes the barrier function $B(x)$. So, the inequality constraints from the optimization problem is now included in the objective function to be minimized.

$$\min_{x \in R^n} f'(Af, m) + \lambda R(f) + \mu_k B(x) \tag{2.5}$$

This method, by introducing a barrier term, converts the inequality constrained problem to an unconstrained problem.

2.3 Primal Dual Method

Let's write the general variational form of the image reconstruction problem from equation 2.4 as:

$$\min_{z \in R^n} F(Az) + G(z)$$

This is referred to as the primal problem, as used in [5].

The convex conjugate of F is :

$$F^*(y) = \sup_{z \in D} \left(\langle y, Az\rangle - F(Az)\right)$$

where D is the domain of F.

We notice from the conjugate function, that it satisfies the Fenchel's inequality [4]:

$$F(z) + F^*(y) \geq \langle y, z\rangle$$

The dual optimization problem can then be written as:

$$\max_{y} -(F^*(y) + G^*(-A^*y))$$

Now,this can be written as a saddle point problem by using duality as:

$$\min_{z \in R^n} \max_{y} \langle y, Az\rangle + G(z) - F^*(y)$$

At the optimal solution, the primal dual gap is zero:

$$F(Az) + G(z) + F^*(y) + G^*(-A^*y) = 0$$

Algorithm 1 represents this method, where for each iteration, the proximal point is determined for both primal and dual problem, followed by the extrapolation of the primal proximal.

[2] defines the proximity operator as:

Definition 2.3.1. For every $f \in \Gamma_0(R^n)$, $x \in R^n$, the proximity operator is defined as:

$$prox_{\gamma f} : R^n \to R^n = \arg\min_{u \in R^n} \frac{1}{2} \parallel x - u \parallel^2 + \gamma f$$

where $\gamma \in R_{>0}$, and Γ_0 denotes a convex functional.

The proximity operator can be thought of as a trade off between minimizing f, and being close to x.

Algorithm 1: Primal Dual Algorithm [5]

For $\gamma, \sigma > 0, \gamma\sigma\|A\| < 1$
$y_{k+1} = prox_{\gamma_{F^*}}(y_k + \gamma A\hat{z}^k)$
$z_{k+1} = prox_{\sigma_G}(z_k - \sigma A^T y^{k+1})$
$\hat{z}^{k+1} = 2z^{k+1} - z^k$

2.4 Forward Backward Method

This method introduces a gradient step at each iteration, and hence the name 'forward-backward' method [3]. Equation 2.5 is addressed by the forward-backward method which deals with $h(f, m, \lambda) = f'(Af, m) + \lambda R(f)$ and $\mu B(x)$ separately, by alternatively iterating through a forward gradient step on the former term and a backward proximal step on the latter. Algorithm 2 depicts the method.

Algorithm 2: Forward-Backward proximal IPM

Let f_0 belong to the feasible region, $\gamma > 0$ and $(\gamma_k)_{k \in N}$ such that $(\forall k \in N)\gamma \leq \gamma_k$;
for *k = 0,1,....* **do**
  do $f_{k+1} = prox_{\gamma_k \mu_k B}(f_k - \gamma_k \nabla_1 h(f_k, m, \lambda)$
end

3. Machine Learning

A machine learning system is the one which is trained for a large amount of input and output sets known as training sets, based on which the model calculates the necessary parameters, so that for a different set of data, the parameters are such that it gives the expected output. It basically learns as it gets trained. A classical programming algorithm takes in rules and data as input, applies those rules to the data to give us the results. A Machine learning algorithm on the other hand takes in the data and the results as a training set and learns the rules.

The reference [16] defines machine learning as a set of methods that can automatically detect patterns in data, and then use the uncovered patterns to predict future data, or to perform other kinds of decision making under uncertainty.

The applications of Machine Learning range from speech recognition and image classification to medical diagnosis, to name a few.

Machine learning is mainly divided into two main types:

1. **Supervised learning**: As the name suggests, it consists of training the model inputs with known targets. The algorithm learns to map these inputs with the help of outputs known while training. The learning model for image deblurring is supervised.

2. **Unsupervised learning**: This involves learning the patterns and transformations in the input data without knowing any targets. This is used for the purpose of data visualization or data analytics.

A machine learning model works on three sets: the training set, validation set and the final test set. The training set is the one on which the model trains itself, the validation set is used to check the performance of the model, as a feedback while training, and the test set is the one on which we test if the model works as expected. The test set should be completely different from the training and validation sets.

3.1 Neural Network and Deep Learning

Deep learning is a Machine learning method that trains the system by filtering data from many layers. Let us first understand how Artificial neural network works.

In [12], D.Kriesel defines Neural network as:

Definition 3.1.1. A neural network is a sorted triple (N, V, w) with two sets N,V and a function w, where N is the set of neurons and V a set $\{(i,j)|i,j \in N\}$ whose elements are called connections between neuron i and neuron j, denoted as w_{ij}. It is either undefined or zero for connections that do not exist in the network.

The neuron is activated or switched on when the output exceeds a threshold value, which is determined by the activation function. This can be represented as:

$$y(x) = g(\sum_{i=1}^{n} w_i x_i - \varepsilon) = g(w^T x - \varepsilon)$$

where $x \in R^n$ is the input vector, $w \in R^n$ is the weights vector, $y \in -1, 1$ is the output of the neuron, $\varepsilon \in R$ is the threshold value, and g is the neuron activation function, which is taken to be the sign function:

$$g(t) = sgn(t) = \begin{cases} 1 & t \geq 0 \\ -1 & t \leq 0 \end{cases}$$

Figure 3.1 depicts the structure of neuron and the working of activation function.

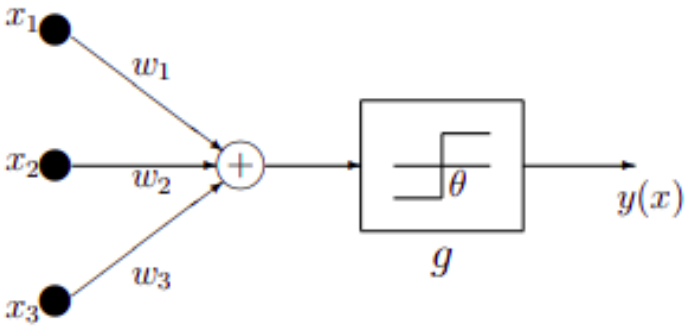

Figure 3.1: Neuron structure described in [8]

In the model, the vertices acts as neurons transporting data and the edges have weights to assign them. Now the input data is passed through the neurons and edges to get a predicted value which is then compared with the actual output to adjust the weights. The model is trained this way for many sets of data to get a higher accuracy of weights. Deep learning is an Artificial Neural Network with many layers, and is analogous to the way Brain filters data. These layers are referred as the depth of the model, therefore the name 'deep' learning.

3.1.1 Deep Feedforward Neural Network

From [7], Deep feedforward network or Multilayer Perceptrons(MLP) aims to approximate some function f^*. If we have a set of inputs as x, and its corresponding output/label y. Now, f^* is maps these two as: $y = f^*(x)$. MLP now defines this mapping by learning the values of the parameters θ in $y = f(x;\theta)$ that would result in the best approximation of the function f^*. It has no feedback connections, since no output models are fed back into itself in these networks.

These networks are formed by composing together different functions, $f(x) = f^{(3)}(f^{(2)}(f^{(1)}(x)))$, where $f^{(1)}$ is the first layer of the network, $f^{(2)}$ is the second layer and so on. The model's depth is defined by the length of this function's chain, and the final layer is the output layer. The intermediate hidden layer's dimensions are the width of the model. The first layer is the input vector $x \in R^n$ representing the n input nodes. This is followed by k hidden layers, with each layer being a function of the previous one. The output from previous layers, is received as input by the neurons, which then goes through the activation function, and the outputs then goes on to the next layer as inputs. This goes on till the output layer. MLP is fully connected, since each neuron of a layer is connected to each neuron of the next layer.
Figure 3.2 represents the MLP structure with w_{ij} being the weight of each path between the neurons.

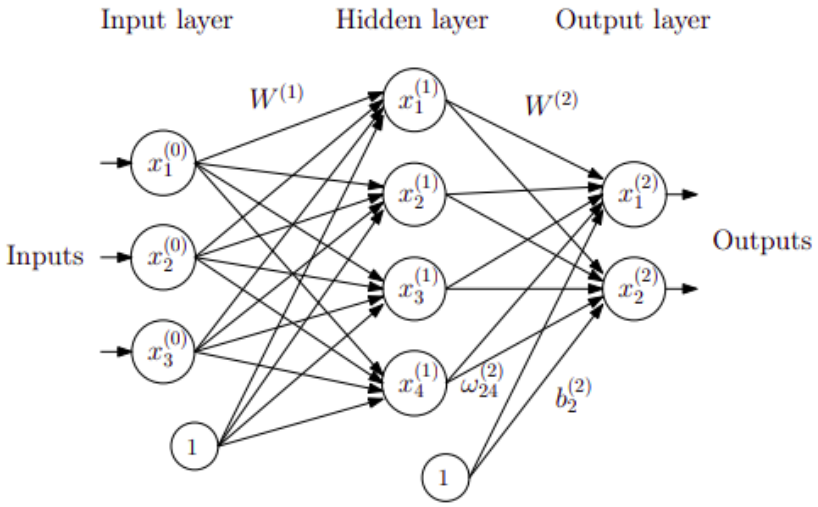

Figure 3.2: Feedforward neural network with depth = 2. Figure from [3]

$$x_j^{(k)} = g^{(k)}(\sum_{i=1}^{d_{k-1}} w_{ji}^{(k)} x_i^{(k)} + b_j^{(k)})$$

for $k = 1, ..., K$ and $j = 1, ..., d_k$ where d_k is the number of neurons in the k^{th} layer.

$$x^{(k)} = g^{(k)}(W^{(k)} x^{(k-1)} + b^{(k)})$$

where $W^{(k)} \in R^{d_k \times d_{k-1}}$ is the weight matrix of k^{th} layer and $b^{(k)} \in R^{d_k}$ is the bias vector.

While training the model, we provide a training set of many input-output pairs $(\{(x_1, y_1), ..., (x_k, y_k)\}$. The deep learning algorithm then modifies the weight parameters to get the correct output, for the input provided.

3.1.2 Convolutional Neural Network

A Convolutional Neural Network is a Deep feedforward neural network which processes data with grid like topology [7]. CNN analyzes images by representing them as an array of pixels and processing the array. But CNNs aren't fully connected, since a huge number of parameters can lead to overfitting. CNNs use sparse connection wherein each neuron in a layer is only connected to a local region of the previous layer. This reduces the number of parameters, and hence avoiding overfitting.
CNN architechture has the following types of layers:

1. **Convolutional layer**.

   This layer involves the convolution of input and the kernel to produce feature map. In a 2D convolution layer, the kernel convolves with the local region of the input layer by sliding through the length and breadth of the layer, producing a weighted sum corresponding to each region. This is then passed through the activation function like rectified linear unit, or ReLU (Figure 3.3), which is defined as:

   $$g(x) = \max(0, x)$$

   This gives us the feature map.

   Figure 3.4 represents the function of the convolutional layer.

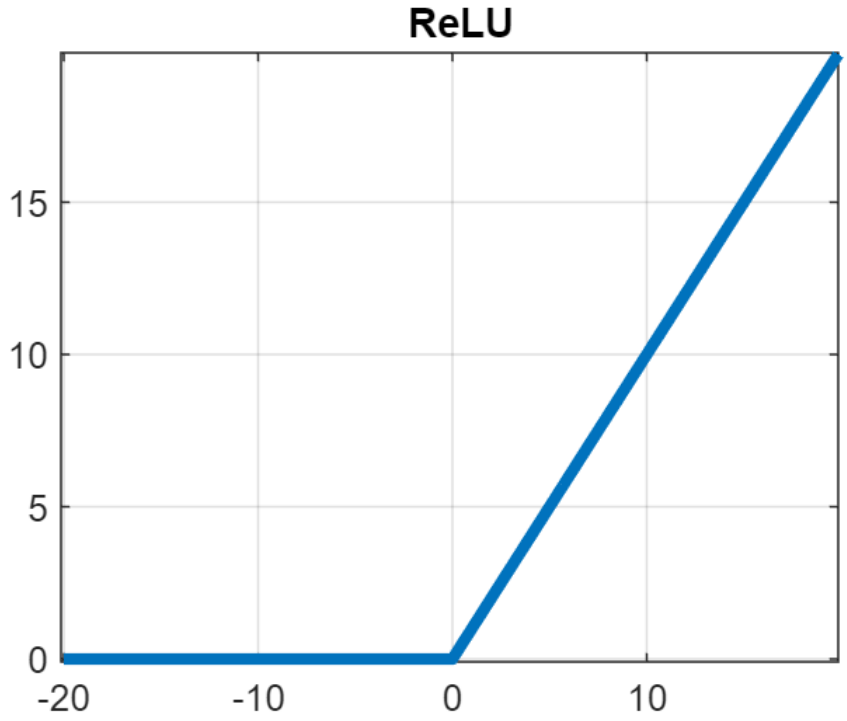

Figure 3.3: The ReLU activation function

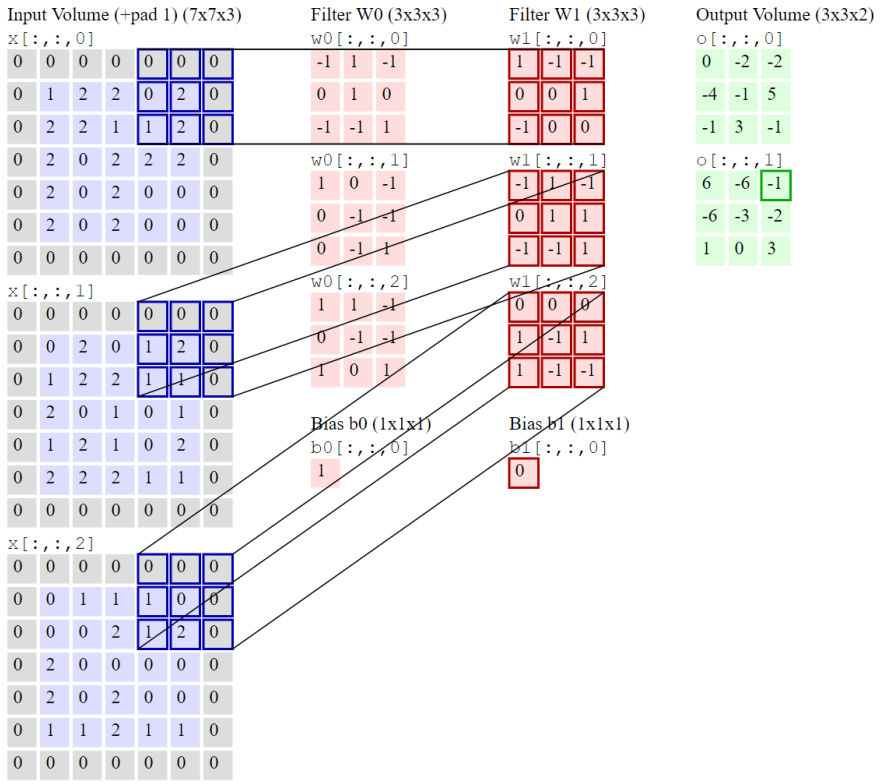

Figure 3.4: 2D Convolution layer example from [13]

2. **Pooling layer**

Now, having a large number of parameters in the network could be computationally expensive and could also lead to over-fitting. The pooling layer is introduced between successive convolution layers, to tackle this problem. This layer reduces the number of parameters, and also the amount of computation. This works on each feature maps obtained from the previous step separately.

For example, in case of max pooling, the maximum from all the elements of the feature map in the pooling filter is selected, and the rest of the computation would continue on this selected feature map. Similarly, in case of average pooling, the average of all the elements of the feature map in the pooling filter are calculated to get a new feature map with fewer parameters.

[13] gives a general summary of the pooling layer as:

(a) Accepts a volume of size $W_1 H_1 D_1$

(b) Requires two hyperparameters:

   i. their spatial extent F,

   ii. the stride S,

(c) Produces a volume of size $W_2 H_2 D_2$ where:

   i. $W_2 = (W_1 F)/S + 1$

   ii. $H_2 = (H_1 F)/S + 1$

iii. $D_2 = D_1$

(d) Introduces zero parameters since it computes a fixed function of the input.

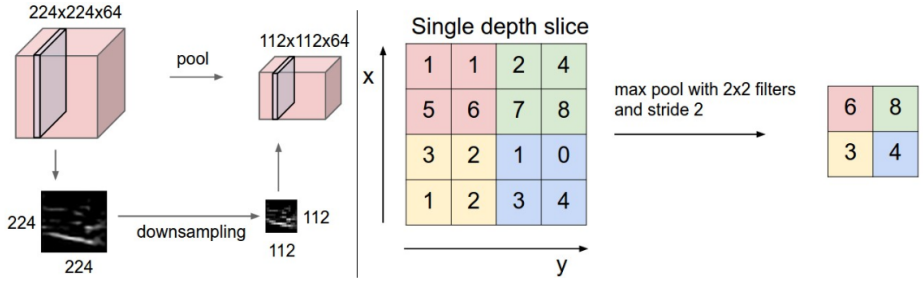

Figure 3.5: Pooling layer downsamples the volume spatially, independently in each depth slice of the input volume. Left: In this example, the input volume of size $[224 \times 224 \times 64]$ is pooled with filter size 2, stride 2 into output volume of size $[112 \times 112 \times 64]$. Notice that the volume depth is preserved. Right: The most common downsampling operation is max, giving rise to max pooling, here shown with a stride of 2. That is, each max is taken over 4 numbers (little 2×2 square). Image from [13]

3. **Fully Connected layer**

Neurons in a fully connected layer have full connections to all activations in the previous layer, as seen in Section 3.1.1

4. Materials and Methods

4.1 The iRestNet Architecture

[14] proposes the iRestNet architechture, which iterates the interior point method using the forward-backward approach described in Algorithm 2 for a finite number of iterations.

The method is unfolded over K iterations, to get the optimal parameters by training the model through supervised learning.

$$f_{k+1} = A(f_k, \mu_k, \gamma_k, \lambda_k)$$

where

$$A(f_k, \mu_k, \gamma_k, \lambda_k) = prox_{\gamma_k \mu_k B}(f_k - \gamma_k \nabla_1 h(f_k, m, \lambda_k)$$

$\forall\ k \in \{0, ..., K-1\}$

Every k^{th} layer L_k is built as the association of three hidden structure $L_k^{(\mu)}$, $L_k^{(\gamma)}$, and $L_k^{(\lambda)}$ to infer the barrier parameter μ_k, the stepsize γ_k and the regularization parameter λ_k respectively. This is then followed by the next iteration A. After going through all the layers, it goes through a final post processing layer L_{pp}. Figure 4.1 shows the iRestNet Architecture.

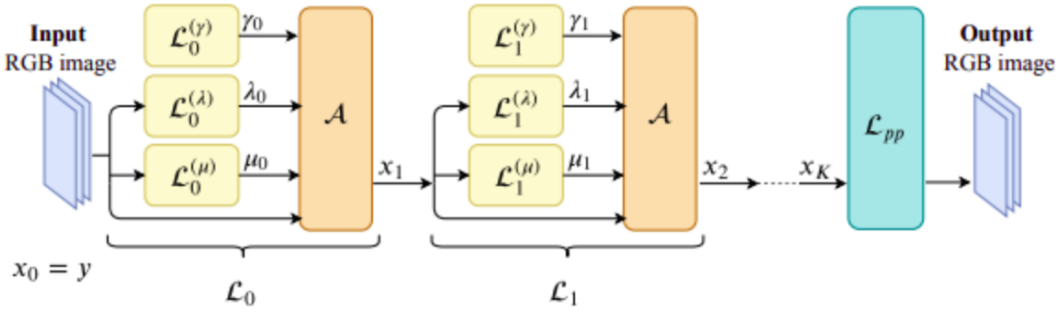

Figure 4.1: iRestNet Architecture. Image from [14]

Now for every k, the output of these hidden structures $(\mu_k, \gamma_k \lambda_k)$ must be positive, which is achieved by the softplus activation function defined as:

$$g(t) = \ln(1 + e^t) \quad (\forall t \in R)$$

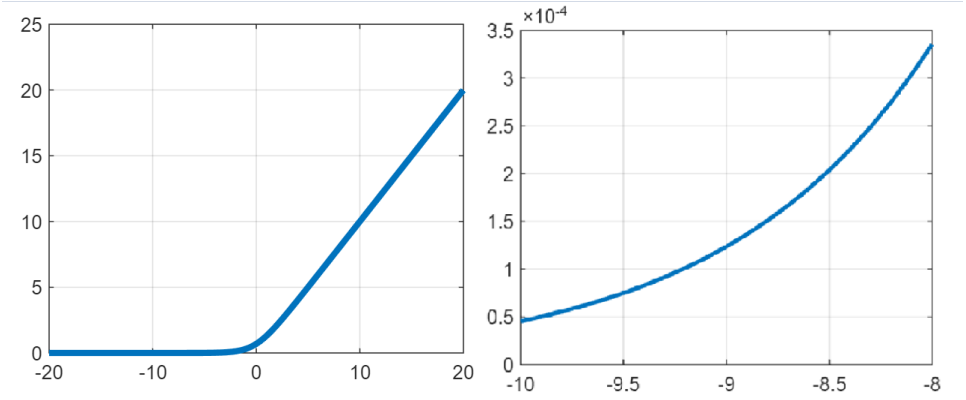

Figure 4.2: Left: The Softplus activation function Right: A closer look at the Softplus activation function. It shows the gradient of this function is never completely zero

Figure 4.2 depicts the activation function. It is similar to ReLU function and is more smooth. A closer look in Figure 4.2 tells us that unlike the ReLU function, the gradient of Softplus is never completely zero, which helps in propagating the gradient through network.

The hidden structures are computed as:

- **Stepsize**

  The stepsize is estimated as:

  $$\gamma_k = L_k^{(\gamma)} = \text{Softplus}(a_k)$$

  where a_k is a scalar parameter that is learned during training.

- **Barrier parameter**

  The barrier parameter μ_k is obtained using two convolutional layers followed by average pooling layers, followed by fully connected network. The architecture of the barrier parameter is shown in Figure 4.3.

- **Regularization parameter**

  The regularization parameter λ_k depends on the regularizational functional R from equation 2.3

The architecture of the post processing L_{pp} layer [20] (shown in Figure 4.5) is made of 9 convolutional layers, with size 3×3. ReLU activation function is used after each convolution step, and the activation function at the final step is the sigmoid function which is defined as:

$$g(x) = \frac{1}{1 + e^{-x}}$$

The sigmoid function is shown in Figure 4.4

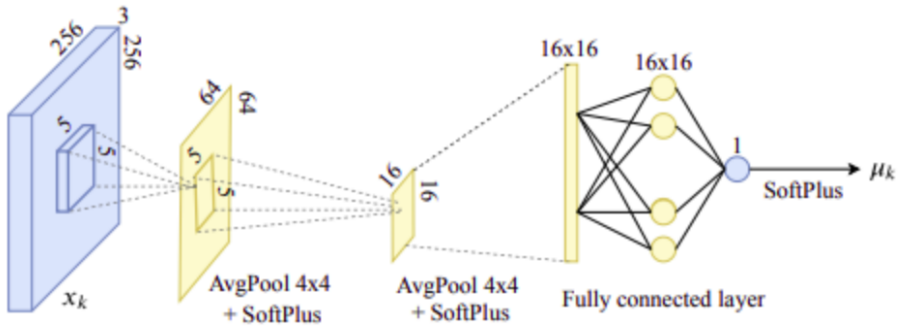

Figure 4.3: Architecture of $L_k^{(\mu)}$. Image from [14]

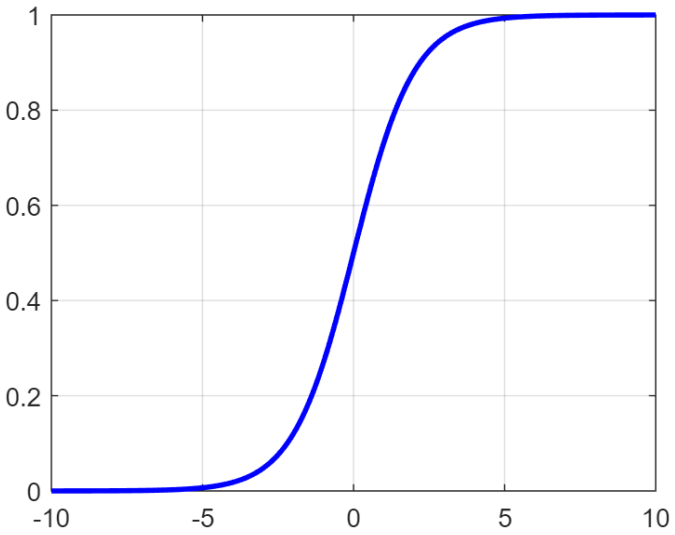

Figure 4.4: Sigmoid Function

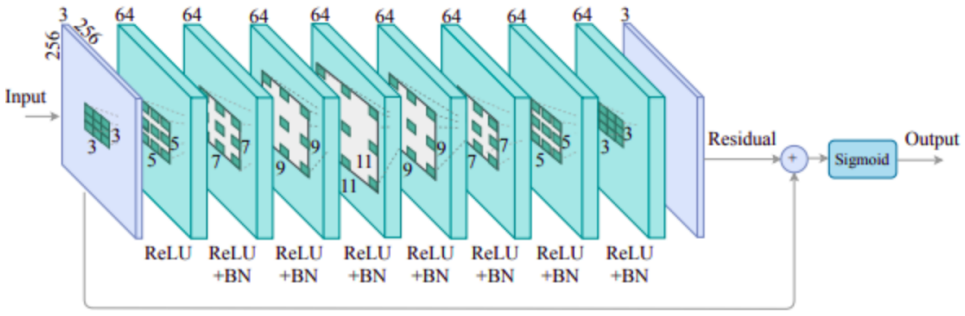

Figure 4.5: Architecture of L_{pp}. Image from [14]

4.2 Total Variation Regularization

The total variation algorithm is implemented using the FlexBox tool [5].

This tool uses the Primal Dual algorithm mentioned in Chapter 2. The Total Variation deblurring is implemented using Rudin-Osher-Fatemi model [18].

The primal optimization problem, for image reconstructions is:

$$\arg\min_{u} \frac{1}{2}\|f - m\|_2^2 + \alpha\|\nabla f\|_{1,2}$$

where m is the input image, and f is the unknown (original image) being fit to m and the second term is the Total Variation term, with α being the regularization parameter.

This tool uses primal dual residual proposed by Goldstein, Esser and Baraniuk [6] as a stopping criterion.

$$p^k = |\frac{z^k - z^{k+1}}{\gamma} - A^T(y^k - y^{k+1})|$$

$$d^k = |\frac{y^k - y^{k+1}}{\sigma} - A(z^k - z^{k+1})|$$

where p^k and d^k are the primal and dual residual respectively, and $|.|$ denotes absolute values. The tool calculates the residual after a fixed number of iterations, which is then scaled with the size of the optimization problem.

4.3 Assessment of the Reconstructed image

Since we are comparing the reconstructed image quality of the iRestNet architechture and Total Variation Regularization, it is important to consider which parameter to use for the analysis. Mean Squared Error and Structural Similarity Index are the two parameters considered in this thesis,and the deblurred images are analysed for these two image measures.

4.3.1 Mean Squared Error

The Mean squared error can be used as an estimate to measure the relative error between two signals. In this case, we measure the error between the original image and the blurred/restored image. [19] defines the Mean sqaured error as:

$$MSE_{(x,y)} = \frac{1}{N}\sum_{i=1}^{N}(x_i - y_i)^2 \tag{4.1}$$

where $x = \{x_i | i = 1, 2, ..., N\}$ and $y = \{xy_i | i = 1, 2, ..., N\}$ are two finite length discrete signals (images in our case), and N is the total number of signal samples (or pixels), and x_i and y_i are the i^th value of sample of x and y respectively.

This measure is then used to calculate the peak signal to noise ratio

$$PSNR = 10log_{10}\frac{L^2}{MSE} \tag{4.2}$$

where L is the range of allowable pixel intensity (255, in our case of gray-scale images).

MSE is the conventional method to calculate the error in the signal, however it isn't an accurate measure on the similarity/differences of the signals as perceived by the human eye. For example, if the spatial arrangement of the signals is changed the MSE from equation 4.1, remains the same, despite the image now being considerably different to the person looking at it. Figure 4.6 depicts this scenario.

4.3.2 Structural Similarity Measure

To assess the quality of the reconstructed image, we use the structural similarity measure [21], which compares the structural dependencies of the pixels of the original and the restored or blurred image. This measure works on the principle that human eye is more sensitive to structural information in images, and comparing the structural changes is an important part to measure the similarities/differences between them. Figure 4.7 shows the sensitivity to structural distortions.

The SSIM measurement is based on three factors:

- The similarity in the luminances.
- The similarity in contrasts.
- The similarity in the structures.

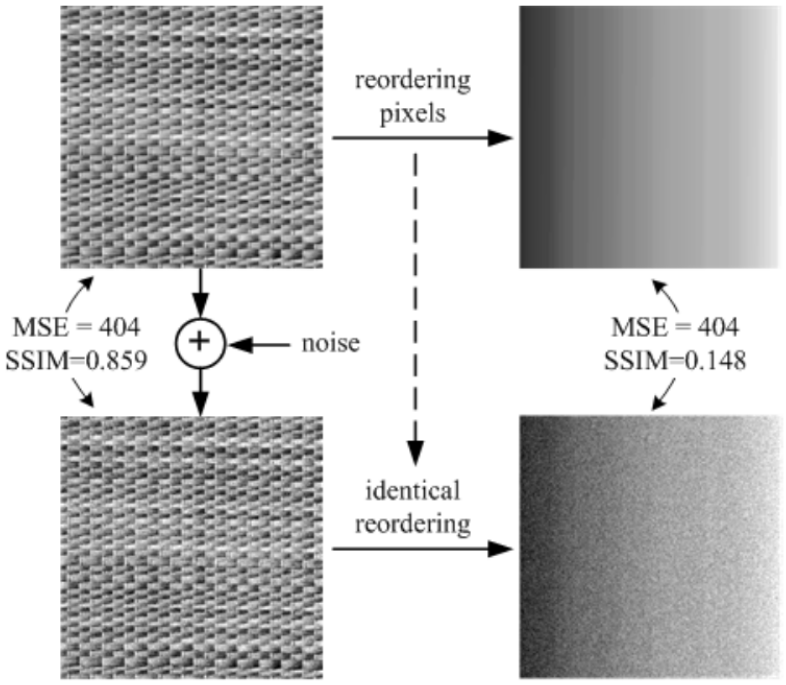

Figure 4.6: MSE is unchanged on reordering the pixels. Image from [19]

This is represented in [21] as:

$$S(x,y) = l(x,y)^{\alpha} c(x,y)^{\beta} s(x,y)^{\gamma} \tag{4.3}$$

where

$$l(x,y) = (\frac{2\mu_x\mu_y + C_1}{\mu_x^2 + \mu_y^2 + C_1})$$

$$c(x,y) = (\frac{2\sigma_x\sigma_y + C_2}{\sigma_x^2 + \sigma_y^2 + C_2})$$

$$s(x,y) = (\frac{\sigma_{xy} + C_3}{\sigma_x\sigma_y + C_3})$$

$\alpha > 0$, $\beta > 0$ and, $\gamma > 0$ define the relative importance of these components.

μ_x and μ_y are the local means, and σ_x and σ_y are the local standard deviations of x and y respectively, and σ_{xy} is the cross correlation of x and y. $C_1 > 0$, $C_2 > 0$, and $C_3 > 0$ are stabilizing terms. Figure 4.8 depicts the SSIM measurement model from these three components.

SSIM is measured in range $0 - 1$, where a score of 0 indicates, no structural similarity, whereas a score of 1 indicates perfect structural similarity.

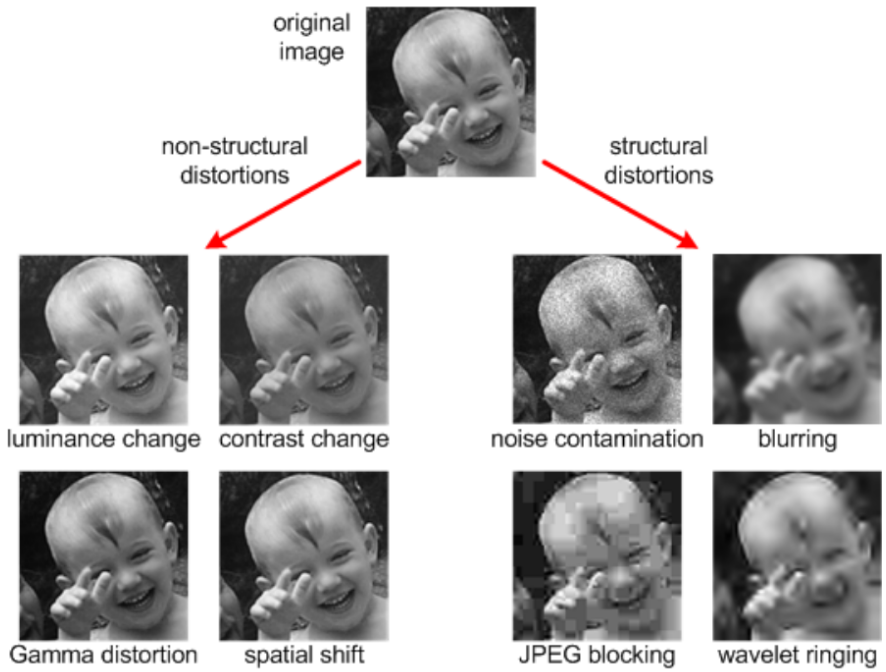

Figure 4.7: Structural and non-Structural distortions. Image from [19]

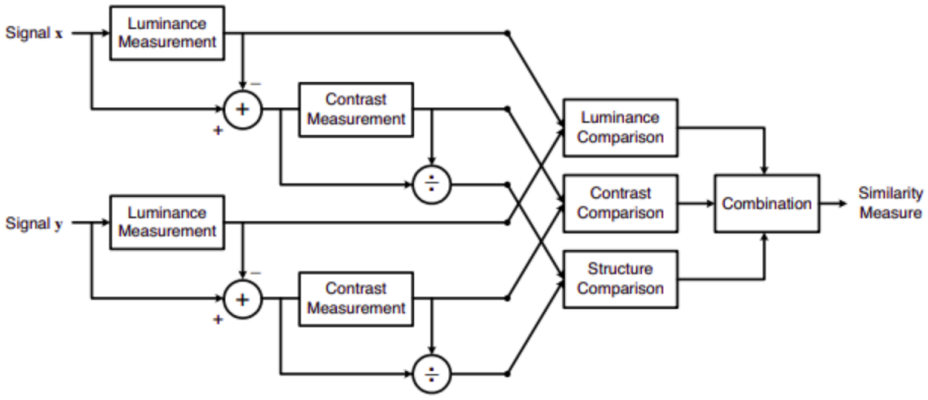

Figure 4.8: SSIM Measurement system. Image from [21]

4.4 Datasets

The trained model from[3] was used, which consist of the Berkeley segmentation training set (BSD500) and COCO training set. The test images consist of the Flickr30 testset, which are cropped to 256×256, and the following point spread function are used of size 25×25:

1. The Gaussian A Kernel with standard deviation of 1.6 pixels and gaussian noise with standard deviation 0.008.
2. The Gaussian B Kernel with standard deviation of 1.6 pixels and gaussian noise with standard deviation uniformly distributed between 0.01 and 0.05.
3. The Gaussian C Kernel with standard deviation of 3 pixels and gaussian noise with standard deviation of 0.04.
4. The Motion blur kernels from [1]
5. A square uniform kernel of size 7×7

The Figure 4.9 shows the kernels used in the model

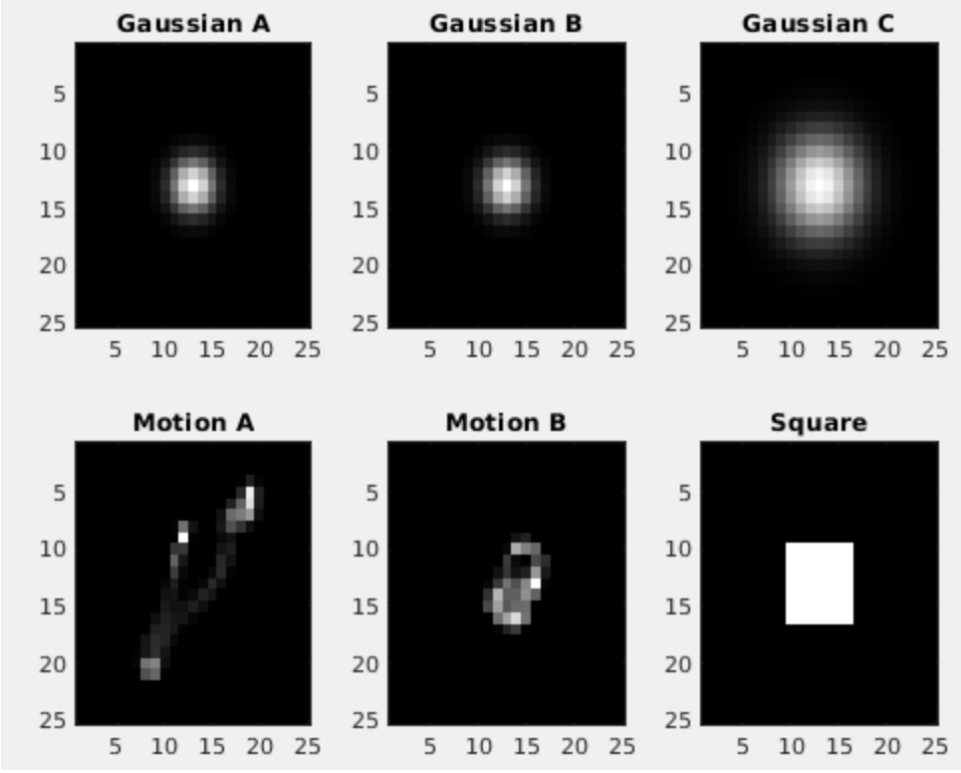

Figure 4.9: Kernels used in the training model of [3]

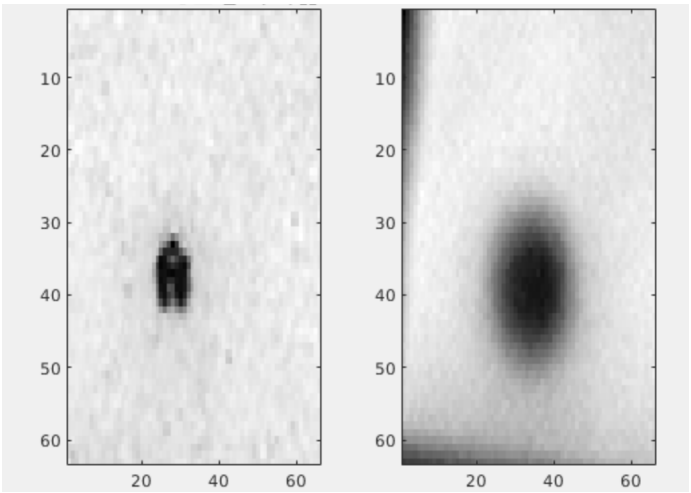

Figure 4.10: The images zoomed in to the black dot to extract PSF Left: Original image Right: Blurred version

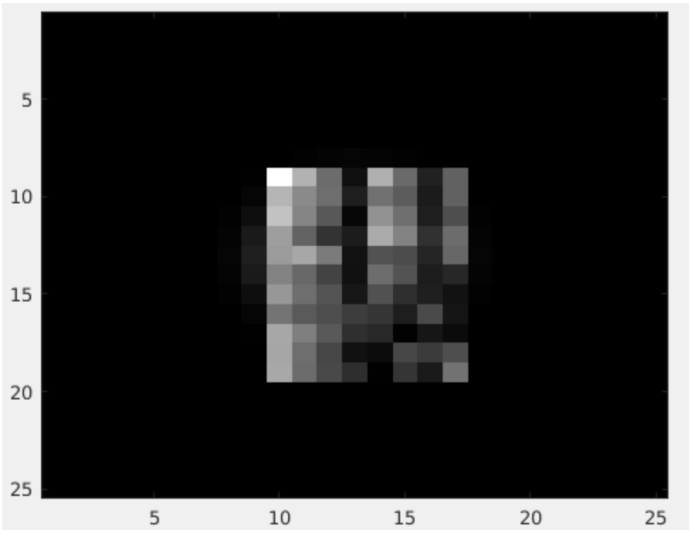

Figure 4.11: The point spread function from fig. 4.10 (Kernel A)

4.4.1 Real Data

The iRestNet model is also trained for the point spread function extracted from real data measurement. This is achieved by taking pictures from different focus setting, to get blurred and original version of the data [17]. The point spread function is then extracted using the blurred and exact versions. The two images are shown in Figure 4.10, and the extracted kernel is shown in 4.11. Let's denote this kernel as 'Kernel A' for our analysis.

5. Results and Discussions

The restored images for the iRestNet and Total Variation method are analysed using both the parameters: Structural similarity index and Mean square error (and Peak signal to noise ratio).

Table 5.1 depict the different values of these indexes for the regularization parameter ranging from $0.0001 - 1$.

The Figures 5.1, 5.4, 5.7, 5.10, 5.13, 5.16, 5.19 depict the restored images from various kernels for both the methods. Figures 5.3, 5.6, 5.9, 5.12, 5.15, 5.18, 5.21 shows the Structural similarity map of the blurred and restored images. The darker spots in the map represents the areas where the distortion between the original image and analysed image is high.

Tables 5.2, 5.3, 5.4 depicts the Average values of Structural similarity measure, Mean Square Error and Peak signal to noise ratio respectively of the iRestNet Neural Network architecture and Total Variation respectively. The averages in all cases are taken form the 30 test images of Flickr30.

The SSIM values indicate that the trained model from iRestNet performs better in all cases than Total Variation, and the difference in performance levels is significantly high for Motion blurs. This is also evident in the SSIM map where the reconstruction from Total Variation has a higher level of darker regions denoting the level of distortion locally in that region compared to the original signal. The regularization parameter gives a better performance for lower values for the considered kernels, in case of Total variation.

Figure 5.1: Left to Right: Original Image from Flickr30, Blurred with Gaussian A kernel (SSIM = 0.721), Restored with iRestNet (SSIM = 0.885), Restored with Total Variation (SSIM = 0.851)

Figure 5.2: Zoomed image of original (Left), iRestNet reconstruction (Middle) and TV reconstruction (Right) for Gaussian A kernel

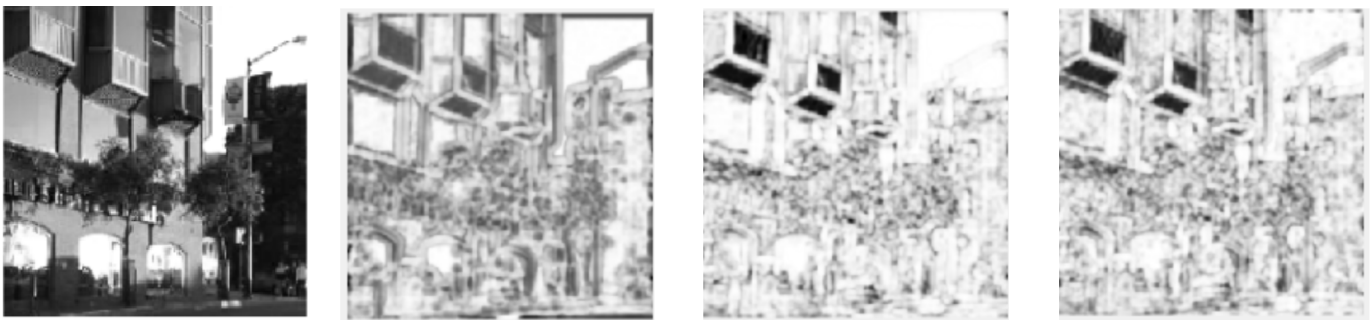

Figure 5.3: Left to Right: Original Image from Flickr30, SSIM map of image blurred with Gaussian A kernel, SSIM map of image restored with iRestNet , SSIM map of image restored with Total Variation

Gaussian A			
Regularization parameter α	Avg SSIM	Avg MSE	Avg PSNR
0.0001	**0.92**	**115.48**	**28.32**
0.001	0.90	148.95	27.61
0.05	0.78	342.81	23.79
0.01	0.85	230.38	25.66
0.5	0.63	791.58	19.64
1	0.58	1122.8	18.01

(a)

Gaussian B			
Regularization parameter α	Avg SSIM	Avg MSE	Avg PSNR
0.0001	0.52	940.62	19.76
0.001	**0.85**	**202.71**	**25.86**
0.05	0.77	344.52	23.74
0.01	0.84	236.33	25.48
0.5	0.63	792.71	19.64
1	0.58	1124.4	18

(b)

Gaussian C			
Regularization parameter α	Avg SSIM	Avg MSE	Avg PSNR
0.0001	0.53	699.54	19.89
0.001	**0.76**	**376.72**	**23.25**
0.05	0.68	545.02	21.46
0.01	0.73	433.47	22.64
0.5	0.59	970.74	18.68
1	0.55	1281.6	17.40

(c)

Motion A			
Regularization parameter α	Avg SSIM	Avg MSE	Avg PSNR
0.0001	0.38	1886.3	15.69
0.001	0.50	1652	16.30
0.05	**0.57**	**1282.4**	**17.41**
0.01	0.55	1451.7	16.87
0.5	0.54	1375.2	17.10
1	0.52	1599.4	16.41

(d)

Motion B			
Regularization parameter α	Avg SSIM	Avg MSE	Avg PSNR
0.0001	0.39	1537.3	16.72
0.001	0.54	1262.4	17.62
0.05	**0.63**	**869.72**	**19.16**
0.01	0.62	1002.9	18.56
0.5	0.58	1002.8	18.52
1	0.55	1263	17.46

(e)

Square			
Regularization parameter α	Avg SSIM	Avg MSE	Avg PSNR
0.0001	0.84	150.25	26.67
0.001	**0.89**	**142.29**	**27.58**
0.05	0.73	422.788	22.73
0.01	0.80	295.34	24.53
0.5	0.61	853.60	19.29
1	0.57	1174.4	17.80

(f)

Kernel A			
Regularization parameter α	Avg SSIM	Avg MSE	Avg PSNR
0.0001	0.39	3483.1	12.99
0.001	0.60	3008.7	13.62
0.05	0.59	3040.8	13.56
0.01	**0.61**	**2970.4**	**13.68**
0.5	0.53	3559.9	12.81
1	0.51	3946.7	12.33

(g)

Table 5.1: SSIM, MSE and PSNR values for a range of α

SSIM Values			
Kernel	Blurred	iRestNet	Total Variation
Gaussian A	0.82	0.94	0.92
Gaussian B	0.75	0.90	0.85
Gaussian C	0.60	0.80	0.76
Motion A	0.52	0.96	0.57
Motion B	0.72	0.96	0.64
Square	0.718	0.93	0.89
Kernel A	0.57	0.89	0.61

Table 5.2: Average Structural Similarity measure for iRestNet and Total variation (for the best α from Table 5.1)

MSE Values			
Kernel	Blurred	iRestNet	Total Variation
Gaussian A	329.27	106.43	115.48
Gaussian B	360.89	168.56	202.71
Gaussian C	662.24	371.29	376.72
Motion A	1261	54.41	1282.4
Motion B	563.00	49.55	869.72
Square	533.08	96.55	142.29
Kernel A	3122.8	231.26	2970.4

Table 5.3: Average Mean Square Error for iRestNet and Total Variation (for the best α from Table 5.1)

PSNR Values			
Kernel	Blurred	iRestNet	Total Variation
Gaussian A	23.63	28.72	28.32
Gaussian B	23.11	26.75	25.86
Gaussian C	20.35	23.33	23.25
Motion A	17.48	31.28	17.41
Motion B	21.16	31.54	19.16
Square	21.51	28.93	27.58
Kernel A	13.44	24.91	13.68

Table 5.4: Average Peak Signal to Noise Ratio for iRestNet and Total variation (for the best α from Table 5.1)

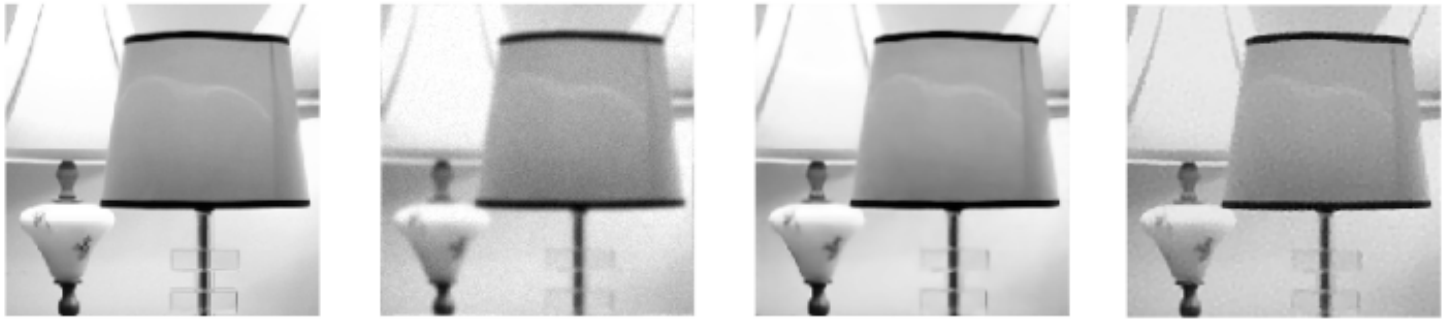

Figure 5.4: Left to Right: Original Image from Flickr30, Blurred with Gaussian B kernel (SSIM = 0.874), Restored with iRestNet (SSIM = 0.982), Restored with Total Variation (SSIM = 0.954)

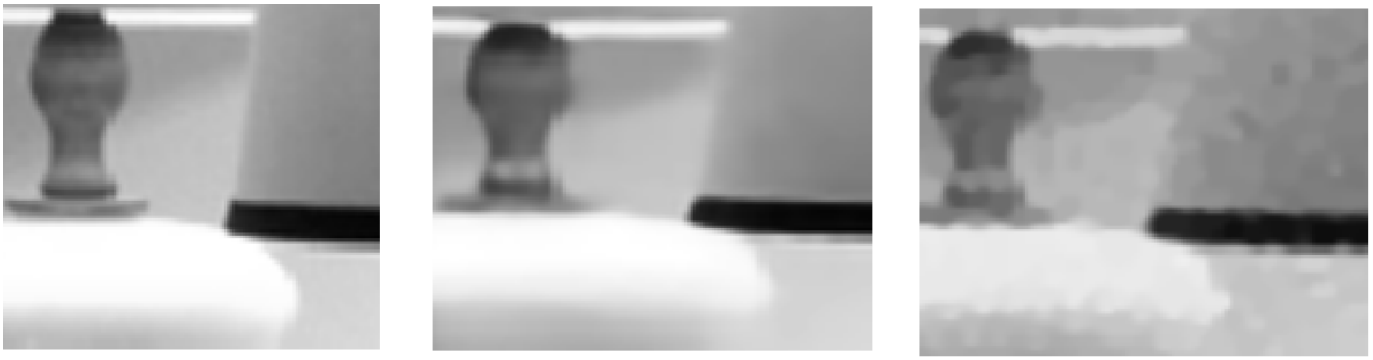

Figure 5.5: Zoomed image of original (Left), iRestNet reconstruction (Middle) and TV reconstruction (Right) for Gaussian B kernel

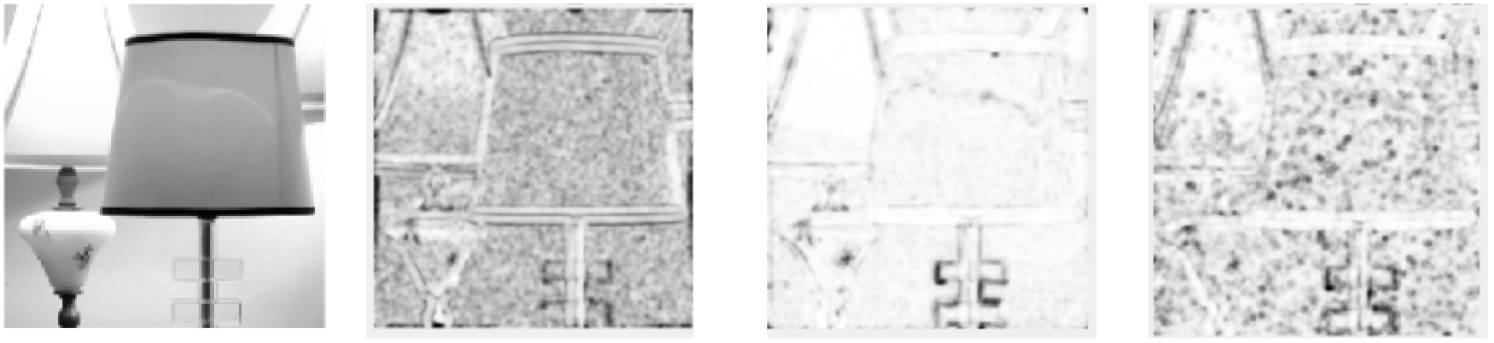

Figure 5.6: Left to Right: Original Image from Flickr30, SSIM map of image blurred with Gaussian B kernel, SSIM map of image restored with iRestNet , SSIM map of image restored with Total Variation

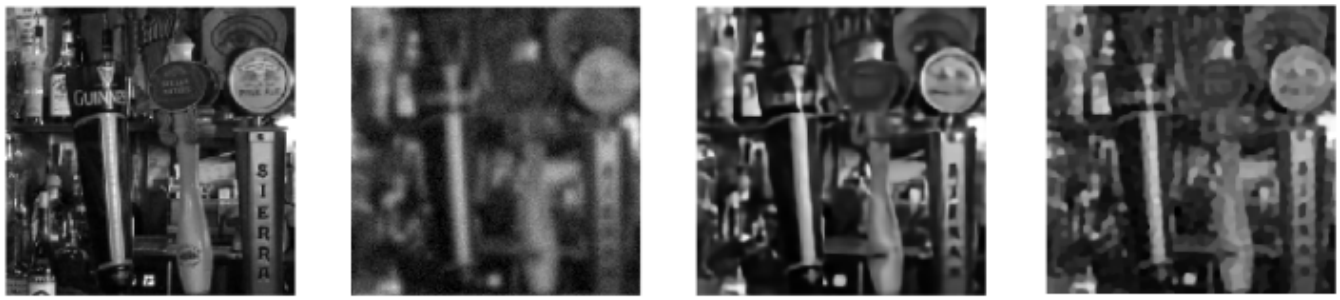

Figure 5.7: Left to Right: Original Image from Flickr30, Blurred with Gaussian C kernel (SSIM = 0.602), Restored with iRestNet (SSIM = 0.793), Restored with Total Variation (SSIM = 0.757)

Figure 5.8: Zoomed image of original (Left), iRestNet reconstruction (Middle) and TV reconstruction (Right) for Gaussian C

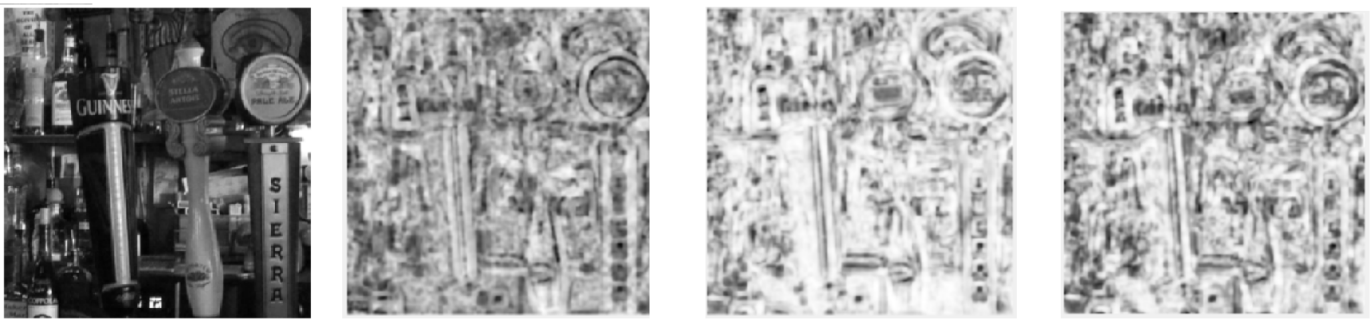

Figure 5.9: Left to Right: Original Image from Flickr30, SSIM map of image blurred with Gaussian C kernel, SSIM map of image restored with iRestNet , SSIM map of image restored with Total Variation

Figure 5.10: Left to Right: Original Image from Flickr30, Blurred with Motion A kernel (SSIM = 0.411), Restored with iRestNet (SSIM = 0.939), Restored with Total Variation (SSIM = 0.455)

Figure 5.11: Zoomed image of original (Left), iRestNet reconstruction (Middle) and TV reconstruction (Right) for Motion A kernel

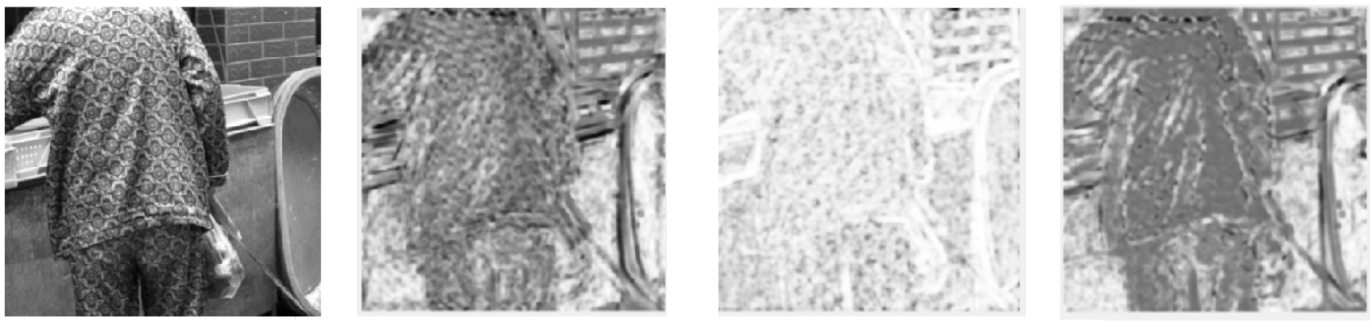

Figure 5.12: Left to Right: Original Image from Flickr30, SSIM map of image blurred with Motion A kernel, SSIM map of image restored with iRestNet , SSIM map of image restored with Total Variation

Figure 5.13: Left to Right: Original Image from Flickr30, Blurred with Motion B kernel (SSIM = 0.815), Restored with iRestNet (SSIM = 0.959), Restored with Total Variation (SSIM = 0.744)

Figure 5.14: Zoomed image of original (Left), iRestNet reconstruction (Middle) and TV reconstruction (Right) for Motion B kernel

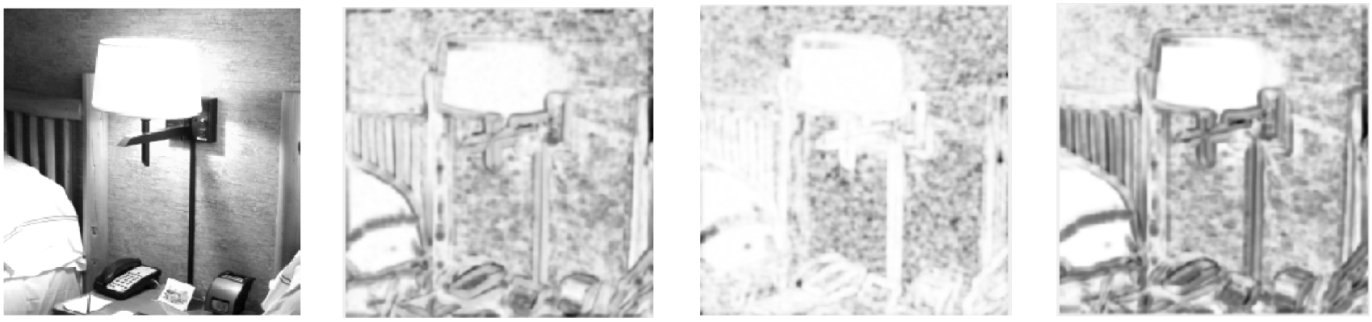

Figure 5.15: Left to Right: Original Image from Flickr30, SSIM map of image blurred with Motion B kernel, SSIM map of image restored with iRestNet , SSIM map of image restored with Total Variation

Figure 5.16: Left to Right: Original Image from Flickr30, Blurred with Square kernel (SSIM = 0.739), Restored with iRestNet (SSIM = 0.956), Restored with Total Variation (SSIM = 0.918)

Figure 5.17: Zoomed image of original (Left), iRestNet reconstruction (Middle) and TV reconstruction (Right) for Square kernel

Figure 5.18: Left to Right: Original Image from Flickr30, SSIM map of image blurred with Square kernel, SSIM map of image restored with iRestNet , SSIM map of image restored with Total Variation

Figure 5.19: Left to Right: Original Image from Flickr30, Blurred with kernel A (SSIM = 0.691), Restored with iRestNet (SSIM = 0.923), Restored with Total Variation (SSIM = 0.733)

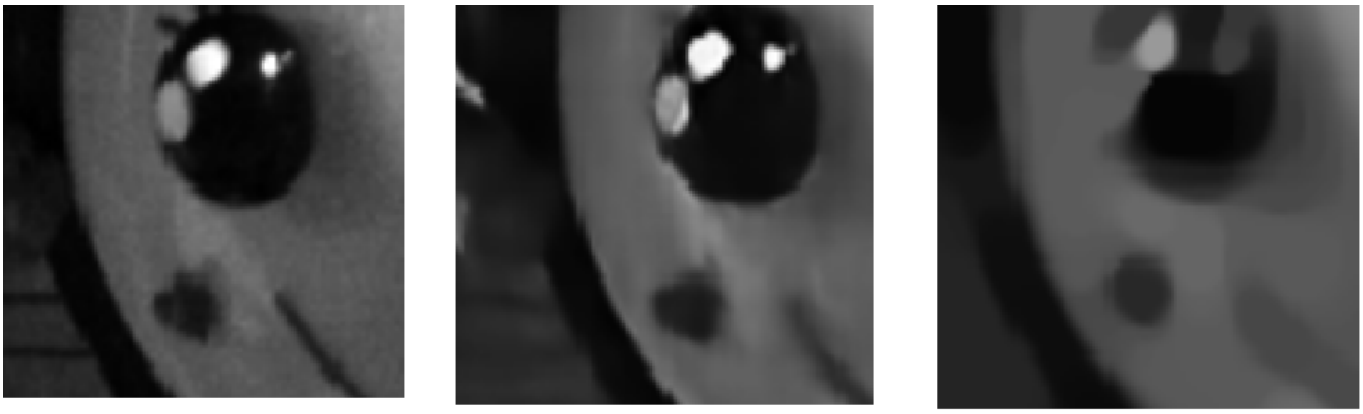

Figure 5.20: Zoomed image of original (Left), iRestNet reconstruction (Middle) and TV reconstruction (Right) for kernel A

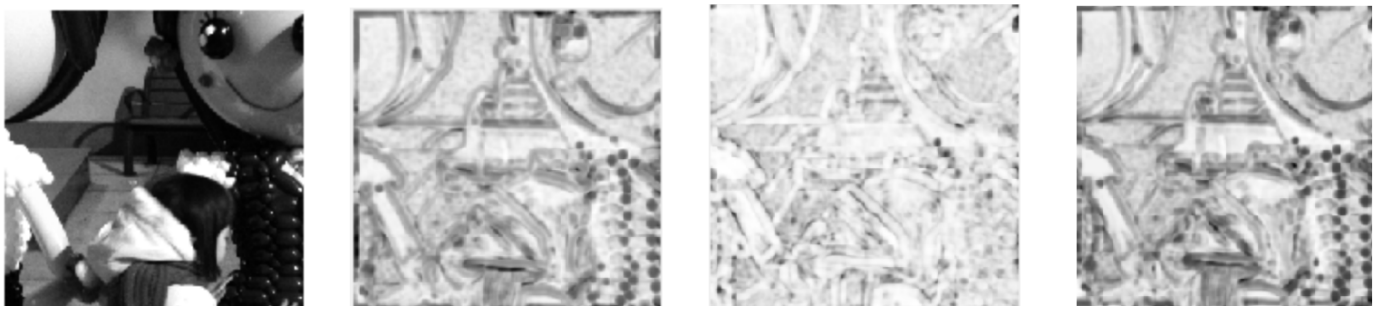

Figure 5.21: Left to Right: Original Image from Flickr30, SSIM map of image blurred with kernel A, SSIM map of image restored with iRestNet , SSIM map of image restored with Total Variation

6. Conclusions

This thesis analyzed the performance of a Deep learning and an analytical method in Deblurring of an image. For this purpose, the iRestNet algorithm [14] was considered for training the model, and the Flexbox tool for Total variation regularization method [5]. We study the interior point approaches behind both these methods, where we want to minimize the variational representation of the image reconstruction problem formalized from the Maximum a posteriori approach. The total variation algorithm is computed for the regularization parameter (α) ranging from 0.0001 to 1, and as the results show, the performance is better for lower range of α and as it reaches closer to 1 (and higher), the result becomes smoother and smoother representing more interaction with the surrounding pixels, and loses its sharpness.

The deblurred images from iRestNet are converted to grayscale to compare them with Total Variation. From the results of Peak Signal to noise ratio, mean square error and Structural Similarity Index, we notice that the learning methods performs better than the analytical method. The Structural similarity map plotted also show that the learning method manages to deblur the image to a much better extent than the Total variation method.

For the Gaussian and Square kernels, we notice that both the iRestNet and Total Variation methods perform quite well. But in case of Motion Blur kernels, the difference in performance of these methods is quite clearly visible, with the iRestNet method performing much better, and the Total variation method gives an almost similar SSIM as the blurred one. For 'Kernel A' [17], which was extracted using blurred and original image of Real data captured, we see that iRestNet method gives a better deblurred output than Total variation.

We notice that Deep learning method in this case performs better than the analytical method in terms of quality of the deblurred output image. But the learning model needs to be trained for every new blur kernel we want to perform the computation for, which takes quite a lot of time unlike the analytical method. On the other hand, Total variation needs to be tuned in for the appropriate regularization parameter which works the best for that kernel, and hence it requires analysis for a range of regularization parameter, while the Deep learning method calculates its parameters during the training of the model.

A. Convex Optimization

Let's understand the convex optimization problem as explained in [4].

Definition A.0.1. A set $C \subset R^n$ is affine if the line through any two distinct points in C lies in C. So, the linear combination of any two points in C would also lie in C, if the coefficients of the linear combination are in normalized form.

Definition A.0.2. A set C is convex if the line segment between any two points in C lies in C.

Definition A.0.3. A function $f : R^n \to R$ is convex, if domain of f is a convex set and if $\forall\ x, y \in$ domain of f, and θ with $0 \leq \theta \leq 1$,

$$f(\theta x + (1-\theta)y) \leq \theta f(x) + (1-\theta)f(y)$$

This means that the line segment between $(x, f(x))$ and $(y, f(y))$ lies above the function f.

A.1 Optimization

Let's define the basic optimization problem with constraints.

$$\begin{aligned} &minimize\ f(x) \\ &subject\ to\ g_i(x) \leq 0,\ i = 1, ..., m \\ &\qquad\qquad h_i(x) = 0,\ i = 1, ..., p \end{aligned} \tag{A.1}$$

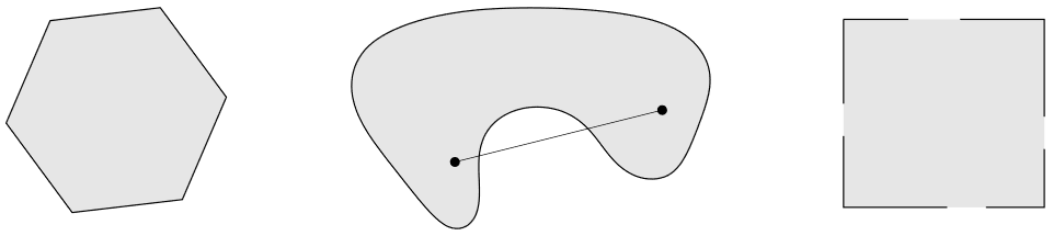

Figure A.1: Left to Right: The hexagon is convex, The kidney shaped set is not convex, since the line segment between the two points in the set shown as dots is not contained in the set,The square contains some boundary points but not others, and is not convex. Image from [4]

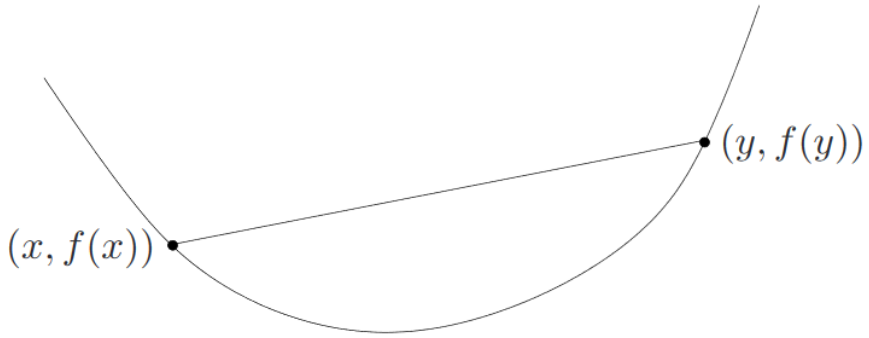

Figure A.2: A convex function, where the line segment between any two points lies above f. Image from [4]

where $x \in R^n$ is the optimization variable, $f : R^n \to R$ is the objective function, $g_i(x)$ are the inequality constraint functions, $h_i(x)$ are the equality constraint functions. This problem is feasible if there is atleast one feasible point (which satisfies all the constraints).

For this optimization problem to be convex, it must follow these conditions.

- The objective function f must be convex.
- The inequality constraint functions $g_i(x)$ must be convex.
- The equality constraint functions $h_i(x)$ must be affine.

Definition A.1.1. The conjugate of function f, $f^* : R^n \to R$ is defined as:

$$f^*(y) = \sup_{x \in D}(y^T x - f(x)) \tag{A.2}$$

where D is the domain of f.

Figure A.3 represents the conjugate of f. Let us now introduce the Lagrangian, $L : R^n \times R^m \times R^p \to R$:

$$L(x, \lambda, \nu) = f(x) + \sum_{i=1}^{m} \lambda_i g_i(x) + \sum_{i=1}^{p} \nu_i h_i(x) \tag{A.3}$$

λ_i, ν_i are the Lagrange multiplier associated with the i^{th} inequality and equality constraints respectively. The vectors λ and ν are the dual variables or Lagrange multiplier vectors associated with the convex optimization problem.

The Lagrange dual function $f' : R^m \times R^p \to R$ is defined as the minimum value of the lagrangian over x:

$$f'(\lambda, \nu) = \inf_{x \in D} L(x, \lambda, \nu) = \inf_{x \in D} f(x) + \sum_{i=1}^{m} \lambda_i g_i(x) + \sum_{i=1}^{p} \nu_i h_i(x)$$

For the optimal value p^* of the convex optimization problem in equation A.1, $f'(\lambda, \nu) \leq p^*$

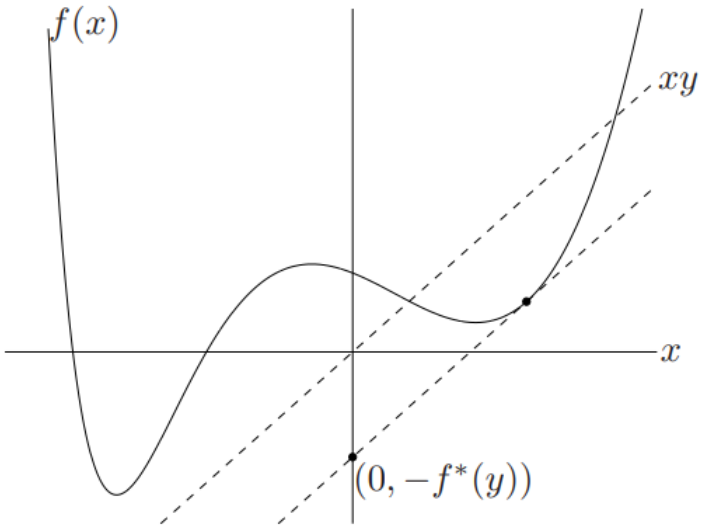

Figure A.3: A function $f : R \to R$, and a value $y \in R$. The conjugate function $f^*(y)$ is the maximum gap between the linear function yx and $f(x)$, as shown by the dashed line. Image from [4]

A.2 Optimality Conditions

Now let us assume that p^* is the optimal value of the primal problem defined in equation A.1, x is primal feasible and (λ, ν) is dual feasible:

$$f(x) - p^* \leq f(x) - f'(\lambda, \nu)$$

where $f(x) - f'(\lambda, \nu)$ is the duality gap. The feasible pair of primal and dual variables $x, (\lambda, \nu)$ localizes the optimal value of the primal and dual problems to an interval of width of the duality gap: $p^* \in [f'(\lambda, \nu), f(x)]$ and $d^* \in [f'(\lambda, \nu), f(x)]$.

For the primal optimal and dual optimal points, the duality gap is zero: $f(x) = f'(\lambda, \nu)$

A.2.1 The Karush-Kuhn-Tucker conditions

Let us now consider primal and dual optimal points with zero duality gap as: x^* and (λ^*, ν^*)

$$\begin{aligned} f(x^*) &= f'(\lambda^*, \nu^*) \\ &= \inf_{x \in D} (f(x) + \sum_{i=1}^{m} \lambda_i^* g_i(x) + \sum_{i=1}^{p} \nu_i^* h_i(x)) \\ &\leq f(x^*) + \sum_{i=1}^{m} \lambda_i^* g_i(x^*) + \sum_{i=1}^{p} \nu_i^* h_i(x^*) \end{aligned} \tag{A.4}$$

Now, since x^* minimizes $L(x^*, \lambda^*, \nu^*)$ over x, its gradient is zero at x^*:

$$\nabla f(x^*) + \sum_{i=1}^{m} \lambda_i^* \nabla g_i(x^*) + \sum_{i=1}^{p} \nu_i^* \nabla h_i(x^*) = 0$$

Now this gives us the Karush-Kuhn-Tucker conditions (KKT conditions):

$$\begin{aligned} g_i(x^*) &\leq 0 \qquad i = 1, ..., m \\ h_i(x^*) &= 0 \qquad i = 1, ..., p \\ \lambda_i^* &\geq 0 \qquad i = 1, ..., m \\ \lambda_i^* g_i(x) &= 0 \qquad i = 1, ..., m \\ \nabla f(x^*) + \sum_{i=1}^{m} \lambda_i^* \nabla g_i(x^*) + \sum_{i=1}^{p} \nu_i^* \nabla h_i(x^*) &= 0 \end{aligned} \tag{A.5}$$

Note that the functions f, $g_1, ..., g_m$, $h_1, ...h_p$ are differentiable.

Now, when the primal optimization problem is convex, then f is convex, g_i is convex, and h_i is affine, then for any points x^*, λ^*, ν^* which satisfies the KKT condition, are the primal and dual optimal points. Hence, the KKT conditions are used to determine if a point in the feasible region is an optimal solution.

Bibliography

[1] A.Levin et al. "Understanding and evaluating blind deconvolution algorithms". In: *IEEE Conference on Computer Vision and Pattern Recognition* (2009), pp. 1964–1971.

[2] Heinz H. Bauschke and Patrick L. Combettes. *Convex Analysis and Monotone Operator Theory in Hilbert Spaces*. Springer.

[3] Carla Bertocchi et al. "Deep unfolding of a proximal interior point method for image restoration". In: (2019). DOI: 10.1088/1361-6420/ab460a.

[4] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004. DOI: 10.1017/CBO9780511804441.

[5] Hendrik Dirks. "A Flexible Primal-Dual Toolbox". In: *ArXiv e-prints* (Mar. 2016).

[6] Tom Goldstein et al. *Adaptive Primal-Dual Hybrid Gradient Methods for Saddle-Point Problems*. 2013. arXiv: 1305.0546 [math.NA].

[7] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[8] L. Grippo and M. Sciandrone. "Metodi di ottimizzazione per le reti neurali". In: *Rapporto Tecnico* 8 (2003), pp. 09–03.

[9] H.W.Engl, M. Hanke, and A.Neubauer. "Regularization of Inverse Problems". In: Kluwer Academic Publishers, Dordrecht, 1996, pp. 48–50.

[10] Per Christian Hansen, James G Nagy, and Dianne P.O'Leary. *Deblurring Images Matrices, Spectra, and Filtering*. Fundamentals of Algorithms. SIAM.

[11] James Joyce. "Bayes' Theorem". In: *The Stanford Encyclopedia of Philosophy*. Ed. by Edward N. Zalta. Spring 2019. Metaphysics Research Lab, Stanford University, 2019.

[12] David Kriesel. *A Brief Introduction to Neural Networks*. 2007. URL: available%20at%20http://www.dkriesel.com.

[13] Fei-Fei Li, Ranjay Krishna, and Danfei Xu. *Convolutional Neural Networks for Visual Recognition*. [Online]. 2020. URL: https://cs231n.github.io/convolutional-networks/.

[14] M.C.Corbineau et al. "Learned Image Deblurring by unfolding a proximal interior point algorithm". In: *IEEE International Conference on Image Processing (ICIP)* (2019).

[15] Jennifer L. Mueller and Samuli Siltanen. *Linear and Nonlinear Inverse Problems with Practical Applications*. Computational Science and Engineering. SIAM. ISBN: 9781611972337.

[16] Kevin P. Murphy. *Machine learning : a probabilistic perspective*. Cambridge, Mass. [u.a.]: MIT Press, 2013. ISBN: 9780262018029 0262018020. URL: https://www.amazon.com/Machine-Learning-Probabilistic-Perspective-Computation/dp/0262018020/ref=sr_1_2?ie=UTF8&qid=1336857747&sr=8-2.

[17] Range of Original and blurred images taken by Samuli Siltanen.

[18] Leonid I. Rudin, Stanley Osher, and Emad Fatemi. "Nonlinear total variation based noise removal algorithms". In: *Physica D: Nonlinear Phenomena* 60.1 (1992), pp. 259–268. ISSN: 0167-2789. DOI: https://doi.org/10.1016/0167-2789(92)90242-F. URL: http://www.sciencedirect.com/science/article/pii/016727899290242F.

[19] Z. Wang and A. C. Bovik. "Mean squared error: Love it or leave it? A new look at Signal Fidelity Measures". In: *IEEE Signal Processing Magazine* 26.1 (2009), pp. 98–117.

[20] Kai Zhang et al. "Learning Deep CNN Denoiser Prior for Image Restoration". In: *IEEE conference on Computer Vision and Pattern Recognition (CVPR)* (2017).

[21] Zhou Wang et al. "Image quality assessment: from error visibility to structural similarity". In: *IEEE Transactions on Image Processing* 13.4 (2004), pp. 600–612.